

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

HardenedBSD

A PROACTIVE SECURITY PROJECT

GETTING TO GRIPS
WITH THE GIMP

HOW TO INSTALL AND
CONFIGURE A WEB SERVER

FINDING OUT ABOUT SYSTEM
CONFIGURATION

WORKING WITH CORE
DUMPS IN GDB

VOL.8 NO.10
ISSUE 10/2014(63)
1898-9144



855-GREP-4-IX
www.ixsystems.com
Enterprise Servers and Storage
for Open Source



- ✓ Rock-Solid Performance
- ✓ Professional In-House Support

FREENAS MINI STORAGE APPLIANCE

IT SAVES YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time.***

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**



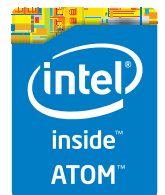
Example of one-bit corruption

The Mini boasts these state-of-the-art features:

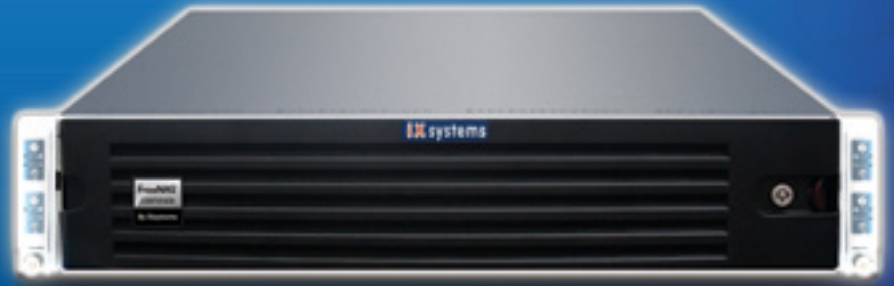
- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured



<http://www.ixsystems.com/mini>



FREENAS CERTIFIED STORAGE



With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

- » Custom built and optimized for your use case
- » Installed, configured, tested, and guaranteed to work out of the box
- » Supported by the Silicon Valley team that designed and built it
- » Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply



<http://www.iXsystems.com/storage/freenas-certified-storage/>

Hello BSD users,

The BSD Mag team is delighted to release the new issue of BSD magazine. This time, a new collection of tutorials and practical articles are included so that you can practice your open source skills. Our ultimate goal is to provide you with the knowledge and skills you need in your professional careers. We'd love to see your suggestions for workshops, tutorials, and topics for upcoming issues. If you think we've missed a very interesting subject that should be covered, do not hesitate to write to us.

Let's take a look at what you will read in this issue of BSD. First, our experts will instruct you on how to install and configure a dedicated web server. Afterwards, we will head to our series of 100+ Unix Commands and Find out About System Configuration. Last but not least, we will learn all about the proactive project for boosting FreeBSD security, HardenedBSD, and much more. Check out this issue for more articles and tutorials.

As always, we would like to send a warm "Thank You" and offer our gratitude to the sources who contributed to this issue and the coming workshops. All the same, we invite other experts for collaboration on future issues; please contact us if interested.

If you want to go on a real life, open source journey with our rich content workshops, publications, tutorials, and so on or if you want to get in contact with our team, please feel free to get in touch with us.

Enjoy reading,
Ewa & the BSD Mag Team

MAGAZINE BSD

Editor in Chief:

Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:

Michael Shirk, Andrey Vedikhin, Petr Topiarz,
Charles Rapenne, Anton Borisov, Jeroen van Nieuwenhuizen,
José B. Alós, Luke Marsden, Salih Khan,
Arkadiusz Majewski, BEng, Toki Winter, Wesley Mouedine
Assaby, Rob Somerville

Top Betatesters & Proofreaders:

Annie Zhang, Denise Ebery, Eric Geissinger, Luca
Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis
Mahangoe, Mani Kanth, Ben Milman, Mark VonFange

Special Thanks:

Annie Zhang
Denise Ebery

Art Director:

Ireneusz Pogroszewski

DTP:

Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:

Paweł Marciniak
pawel@software.com.pl

CEO:

Ewa Dudzic
ewa.dudzic@software.com.pl

Publisher:

Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.

FreeNAS

in an Enterprise Environment

By the time you're reading this, FreeNAS has been downloaded more than 5.5 million times. For home users, it's become an indispensable part of their daily lives, akin to the DVR. Meanwhile, all over the world, thousands of businesses, universities, and government departments use FreeNAS to build effective storage solutions in myriad applications.

What you will learn...

- How TrueNAS builds off the strong points of the FreeBSD and FreeNAS operating systems
- How TrueNAS meets modern storage challenges for enterprise

The FreeNAS operating system is free to the public and offers thorough documentation, an active community, and a feature-rich storage environment. Based on FreeBSD, FreeNAS can share over a host of protocols (SMB, NFS, FTP, iSCSI, etc) and features an intuitive web interface, the ZFS file system, a plug-in system for more features, and much more.

Despite the massive popularity of FreeNAS, many aren't aware of its big brother duties in some of the most demanding environments: the proven, enterprise-grade, professionally-supported line of TrueNAS.

But what makes TrueNAS different? Well, I'm glad you asked...

Commercial Grade Support

When a mission critical storage solution goes down, an organization's whole operation can halt. Whole community-based (and free), it can't always get an immediate response and running in a timely manner. TrueNAS offers the responsiveness and expertise of a dedicated support team to provide that safety.

Created by the same team that developed FreeNAS, TrueNAS is the

DO NOT INTERRUPT THIS MAGAZINE TO BRING YOU THIS IMPORTANT ANNOUNCEMENT:

THE PEOPLE WHO DEVELOP FREENAS, THE WORLD'S MOST POPULAR STORAGE OS, HAVE JUST REVAMPED TRUENAS.



POWER WITHOUT CONTROL MEANS NOTHING. TRUENAS STORAGE GIVES YOU BOTH.

- | | |
|---|--|
| <input checked="" type="checkbox"/> Simple Management | <input checked="" type="checkbox"/> Self-Healing Filesystem |
| <input checked="" type="checkbox"/> Hybrid Flash Acceleration | <input checked="" type="checkbox"/> High Availability |
| <input checked="" type="checkbox"/> Intelligent Compression | <input checked="" type="checkbox"/> Qualified for VMware and HyperV |
| <input checked="" type="checkbox"/> All Features Provided Up Front (no hidden licensing fees) | <input checked="" type="checkbox"/> Works Great With Citrix XenServer® |

To learn more, visit: www.ixsystems.com/truenas



POWERED BY INTEL® XEON® PROCESSORS

Intel, the Intel logo, Intel Xeon and Intel Xeon Inside are trademarks of Intel Corporation in the U.S. and/or other countries.

VMware and VMware Ready are registered trademarks or trademarks of VMware, Inc. in the United States and other jurisdictions.

Citrix makes and you receive no representations or warranties of any kind with respect to the third party products, its functionality, the text(s) or the results therefrom, whether expressed, implied, statutory or otherwise, including without limitation those of fitness for a particular purpose, merchantability, non-infringement or title. To the extent permitted by applicable law, in no event shall Citrix be liable for any damages of any kind whatsoever arising out of your use of the third party product, whether direct, indirect, special, consequential, incidental, multiple, punitive or other damages.

NEW RELEASE



08 HardenedBSD Proactive Security Project

David Carlier

Oliver Pinter and Shawn Webb brought to life a security-centric distribution of FreeBSD called HardenedBSD. HardenedBSD aims to continuously implement kernel and userland hardening features, such as Address Space Layout Randomization (ASLR), protect hardening and many more. In this clause, we hash out all about the HardenedBSD Project illustrated with C++ simple codes.

10 100+ Unix Commands. Finding out About System Configuration Part 2

Craig S. Wright

Some of the other areas that we want to check when testing the security of a *NIX system are detailed in this section. The primary goal should be to take what was learned in the preceding sections and create checklists and scripts to help us initiate this process. In this article, Craig S. Wright presents some of the key objectives in testing a system using the CLI.

24 How to Install a Web Server

Ivan Voras

The goal of this article is to walk the reader through installing and configuring the Apache and PHP servers. This article introduces the roles of Apache and PHP and a step by step guide from configuring the Apache and PHP servers to Configuring a simple web site.

32 Working with Core Dumps in GDB

Carlos Neira

A core dump or core file is produced when a signal (man signal) indicates that it is not possible to continue executing the program because of an error (for example

SIGSEV). A core file contains the current state of the program when it was executed (the stack of each thread, the contents of the CPU registers, the values of global and static variables, etc...). If we are lucky and the program that dumped core was compiled with debug options, the core file will have information about the source code and lines.

40 Getting to Grips With the Gimp – Part 8

Rob Somerville

The Gimp currently supports a number of add-ons: Brushes, Gradients, Patterns, Palettes, Fonts, Scripts and Plug-ins. Adding an extra resource is simplicity itself, once the desired add-on has been found. In this eighth part in our series on the Gimp, we will look deeply at how to extend the Gimp with extra add-ons.

46 The UK government is planning to put trolls in jail for up to two years. Is this a sensible approach in containing the darker side of human nature?

Rob Somerville

Trolling, the deliberate posting of offensive or controversial comments via Forums, Facebook, Twitter etc. is not a new phenomena on the big nasty inter-web. The phrase “Don’t feed the trolls” (DFTT) has been around for quite a while. But what is the psychology behind this desperate and compulsive act? Let’s find out in this article.

CYBER SECURITY EXPO

8-9 October 2014
ExCeL London

A **NEW** event,
for a new era of **cyber threats**

www.cybersec-expo.com

- » The most comprehensive analysis anywhere of how to protect the modern organisation from cyber threats
- » Free to attend seminars delivered by Mikko Hypponen, Eugene Kaspersky and many more
- » Attend the "Hack Den" a live open source security lab to share ideas with White Hat hackers, security gurus, Cyber Security EXPO speakers and fellow professionals
- » Network with industry experts and meet with Cyber Security exhibitors
- » Discover what the IT Security team of the future will look like

→ **Register NOW** 

www.cybersec-expo.com



Cyber Security EXPO is the new place for everybody wanting to protect their organisation from the increasing commercial threat of cyber attacks. Cyber Security EXPO has been designed to provide CISOs and IT security staff the tools, new thinking and policies to meet the 21st century business cyber security challenge.

Cyber Security EXPO delves into business issues beyond traditional enterprise security products, providing exclusive content on behaviour trends and business continuity. At Cyber Security EXPO, discover how to build trust across the enterprise to securely manage disruptive technologies such as: Cloud, Mobile, Social, Networks, GRC, Analytics, Identity & Access, Data, Encryption and more.

Co-located at

IP EXPO EUROPE
8-9 October 2014 ExCeL London

www.ipexpo.co.uk

FREE
REGISTRATION

Sponsors



HardenedBSD Proactive Security Project

Oliver Pinter and Shawn Webb brought to life a security-centric distribution of FreeBSD called HardenedBSD. HardenedBSD aims to continuously implement kernel and userland hardening features, such as Address Space Layout Randomization (ASLR), mprotect hardening, Position Independent Executable (PIE) support, and PTrace hardening, among other features.

The HardenedBSD project started due to the complexity involved in implementing ASLR. I recently joined the HardenedBSD development team, implementing PTrace hardening, arc4random enhancements, `getentropy()` support, and other userland hardening features. In this article, I'd like to detail ASLR. Let's start with this simple piece of sample C++ code (compiled with `-fPIE -pie`): Listing 1.

Executing this sample application multiple times when ASLR is enabled produces this output:

```
address of da 0x9c2406058
address of da 0x8b8806058
address of da 0x9d7806058
```

Since the memory layout of the application is randomized, an attacker will have a hard time determining the location in memory where a given vulnerability might reside. Via `procstat`, you can also see the start address of the concerned binary, for example:

```
procstat -v <pid of the example app>
... First launch
```

Listing 1. Sample C++ code

```
#include <mtld/devatlas.h>
#include <cstdio>
#include <memory>

using namespace std;
using namespace Mobi::Mtld;

int
main(int argc, char *argv[])
{
    ...
    unique_ptr<Da::DeviceAtlas> da(new Da::DeviceAtlas());
    da->loadDataFromFile(argv[1]);
    ...
    printf("addr of da %p\n", da.get());
    ...
    return (0);
}
```



```
PID    START    END ... PATH
54318  0x5afe5000    0x5afe6000  vn /tmp/datest
... Second launch
55773  0x21282000    0x21283000  vn /tmp/datest
...
```

Executing this sample application multiple times when ASLR is disabled produces this output:

```
address of da 0x801c06058
address of da 0x801c06058
address of da 0x801c06058
```

procstat checking

```
... First launch
PID    START    END ... PATH
68978  0x400000    0x401000  vn /tmp/datest
... Second launch
84429  0x400000    0x401000  vn /tmp/datest
```

Note that while ASLR makes life a little bit harder for an attacker, it is not the end-all-be-all of exploit mitigation techniques. Exploit techniques like Blind ROP (BROP) can defeat ASLR. When ASLR is combined with multiple exploit mitigation techniques, security is greatly enhanced and ASLR-defeating exploit payloads can be rendered inadequate.

Conclusions

ASLR is enabled by default for all architectures and controlled by the `PAX_ASLR` kernel option. This means that ASLR will be applied to all supported applications. If a user wishes to disable ASLR for a given application, the user must force that application to opt-out (detailed later).

Another kernel option, `PAX_SYSCTL`, exposes additional tunables (via `sysctl`), allowing ASLR behavior con-

trol without requiring a reboot. By default, the `sysctl security.pax.aslr.status` can only be changed at boot time via `/boot/loader.conf`. Enabling the `PAX_SYSCTL` kernel option allows a root user to modify `security.pax.aslr.status`. See Appendix B for a list of the tunables.

ASLR tunables are per-jail and each jail inherits its parent jail's settings. Having per-jail tunables allows more flexibility in shared-hosting environments. This structure also allows a user to selectively disable ASLR for applications that misbehave. ASLR-disabled applications will still have policy-based security applied to them by virtue of being jailed.

The `mac_bsdextended(4)` MAC module and its corresponding `ugidfw(8)` application have been modified to allow a user to enable *experienceor* or disable ASLR for specific applications. The `filesystem` object specification has been modified to pass the inode along with the filesystem id when the new `paxflags` option is specified. The `paxflags` option is optionally placed at the end of the rule. An upper-case “A” argument to the option signifies ASLR is enabled for the application and a lower-case “a” signifies ASLR is disabled for the application. Sample `ugidfw(8)` rules are in Appendix C. This feature is only implemented in the HardenedBSD[8] project and likely won't be upstreamed to FreeBSD without further review and sign-off from certain key FreeBSD developers.

We are working on upstreaming our ASLR implementation with the FreeBSD project. Sean Bruno is the developer on the FreeBSD side who is sponsoring the commit.

DAVID CARLIER

David Carlier has been working as a software developer since 2001. He used FreeBSD for more than 10 years and starting from this year, he became involved with the HardenedBSD project and performed serious developments on FreeBSD. He worked for a mobile product company that provides C++ APIs for two years in Ireland. From this experience, he became completely inspired to develop on FreeBSD.

100+ Unix Commands.

Learning About System Configuration

Part 2.

Some of the other areas that we want to check when testing the security of a *NIX system are detailed in this section. The primary goal should be to take what was learned in the preceding sections and create checklists and scripts to help us initiate this process. The main reason for incorporating scripts is that we can quickly rerun any test that we have done in the past and we should expect the same (or at least similar) results each time.

Some of the key objectives in testing a system should be to identify and maintain a list of the following information that can be tracked and monitored over time:

- Identify – system type including hardware information and applications
- Identify – patch levels invalidate that these are maintained
- General – system information should also be collected

Operating system version

The command “uname -a” provides processor and operating system information related to the host being evaluated. Commands such as “patchdiag” are useful in analyzing the current versions of applications and software patching.

File systems in use

The “mount” command displays a list of all currently mounted file systems as well as a list of the types of file systems that are mounted.

The “fdisk -l” command is used to validate the mounted partitions against the actual partitions in use on the system.

The “free” command provides information on how much physical memory is installed in the system, how large the

swap partition is, how much space is currently in use and how much swap space is in use.

Reading Logfiles

As was noted above, *NIX log files can be stored in a variety of different locations and formats. Where possible, the aim should be to aggregate and store logs on a remote centralized computer (often called a log server). This host could then be a central location for monitoring many computers on the network. The system could then be firewalled to restrict access and not allow administration from remote sites.

An introduction to logging is freely available in a paper from NIST at <http://csrc.nist.gov/nissc/1998/proceedings/paperD1.pdf>.

What Tools to Use

There are numerous *NIX security tools available and keeping up with changes can be difficult. One of the best sources of information on these tools is available from the “Top 100 Network Security Tools” site maintained by “Fyodor” at <http://sectools.org/>.

NIST also maintains a list of security tools at <http://csrc.nist.gov/tools/tools.htm>.

Password Assessment Tools

Assessing the strength of passwords is an essential task in any *NIX security assessment. There are a number of tools available to do this, including those based on rainbow tables and dictionary-based versions. The definitive “top 10” list for password crackers is again maintained by “Fyodor”. His site, “Top 10 Password Crackers” (<http://sectools.org/crackers.html>) maintains a list of password cracker tools and their availability.

RPCINFO, RPC and Portmapper

*NIX uses “portmap” to register RPC (Remote procedure call) programs. If an application wishes to connect to an RPC-based application, it will first query the portmapper for information about the application. This is done in order to save on low numbered ports.

The portmapper allows multiple ports to be assigned as they are needed. Unfortunately, the portmapper service may be named in a variety of ways. For this reason, it is essential that a checklist is created for your specific system. Many of the aforementioned sites, such as SANS, CIS and NIST, have created comprehensive lists dedicated to a number of operating systems. Portmapper may be designated under *NIX as portmap, rpc.bind, portmapper or several other possibilities.

The portmapper application is actually an RPC program as well. The distinction is that it always listens on ports 111 TCP and UDP. On certain operating systems, such as Solaris, portmapper may also listen on some other high numbered ports. The role of the portmapper service is to provide directory services. These permit applications to register their versions and the port numbers such that applications may query the portmapper to discover if the service is active and which port number it is associated with. This then allows the application to connect to that port.

The tool “rpcinfo” is a standard tool available on practically all varieties of *NIX. The primary commands that the system security tester will need to know include:

- “rpcinfo –p” which is used to discover local services, and
- “rpcinfo –p <target>” which allows the user to discover remote services.

System Accounting and Process Accounting

Accounting reports created by the system accounting service present the *NIX administrator with the information to assess current resource assignments, set resource limits and quotas, and predict future resource requirements. This information is also valuable to the system security

tester and allows for the monitoring of system resourcing. When the system accounting has been enabled on a *NIX system, the collection of statistical data will begin when the system starts or at least from the moment that the accounting service is initiated. The standard data collected by system accounting will include the following categories:

- Connect session statistics
- Disk space utilization
- Printer use
- Process use

The accounting system process starts with the collection of statistical data from which summary reports can be created.

These reports can assist in system performance analysis and offer the criteria necessary to establish an impartial customer chargeback billing system or many other functions related to the monitoring of the system. A number of the individual categories of statistics collected have been listed in the sections that follow.

Connect Session Statistics

Connect session statistics allow an organization to bill, track or charge access based on the tangible connect time. Connect-session accounting data, associated with user login and logout, is composed by the init and login commands. When a user logs into the *NIX system, the login program makes an entry in the “wtmp” file. This file will contain the following user information:

- Date – of login/logout
- Time – of login/logout
- Terminal – port
- User – name

This data may be utilized in the production of reports containing information valuable to both the system security tester and system administrator. Some of the information that can be extracted includes:

- Connect – time seconds used,
- Date – and starting time of connect session,
- Device – address of connect session,
- Login – name,
- Number – of prime connect time seconds used,
- Number – of nonprime connect time seconds used,
- Number – of seconds elapsed from Jan 01st 1970 to the connect-session start time,
- Process – Usage,
- User – ID (UID) associated with the connect-session.

It is also possible to gather statistics about individual processes using system accounting. Some areas that may be collected include:

- Elapsed time and processor time consumed by the process,
- First eight characters of the name of the command,
- I/O (Input/output) statistics,
- Memory usage,
- Number of characters transferred,
- Number of disk blocks read or written by the process,
- User and group numbers under which the process runs.

Many *NIX systems maintain statistical information in a “pacct” or process account database or accounting file. This database is commonly found in the `/var/adm/pacct` file, but like many *NIX log files, this will vary from system to system. The accounting file is used by many of the system and process accounting commands. When a process terminates, the kernel writes information explicitly to the particular process into the “pacct” file. This file consists of the following information:

- Command used to start the process
- Process execution time
- Process owner’s user ID

When system accounting is installed and running on a *NIX system, commands to display, report, and summarize process information will be available. Commands such as “ckpacct” can be used by the administrator or system security tester to ensure that the process accounting file (“pacct”) remains under a set size and thus is stopped from either growing too large or possibly impacting system performance in other ways.

Disk Space Utilization

System accounting provides the ability for the system security tester to receive information concerning the disk utilization of the users. As it is possible to restrict users to a specified disk usage limit, the system security tester may need to validate usage through a disk quota system.

This may be monitored and tested to ensure users are adhering to limits. This allows an unwary client to be charged fees that are correctly associated with another account. Disk usage commands perform three basic functions:

- Collect disk usage by filesystem,
- Gather disk statistics and maintain them in a format that may be used by other system accounting commands for further reporting,
- Report disk usage by user.

Note

It is necessary to be aware that users can avoid charges and quota restrictions for disk usage by changing the ownership of their files to that of another user. The “chown” command provides a simple method for users to change ownership of files. Coupled with the ability to set access permissions (such as through the use of the “chmod” command), a user could create a file owned by another party that they could still access.

Printer Usage

Printer usage data is stored in the “qacct” file (this is commonly located in `/var/adm/qacct` on many systems, though this varies). The “qacct” file is created using an ASCII format. The qdaemon writes ASCII data to the “qacct” file following the completion of a print job. This file records printer queue data from each print session and should, at a minimum, contain the following fields:

- User Name
- User number(UID)
- Number of pages printed

Automatic Accounting Commands

To accumulate accounting data, the *NIX system needs to have a number of command entries installed into the “crontab” file (e.g. the `/var/spool/cron/crontabs/adm` file on many *NIX'es but this will change from system to system). The cron file of the adm user is configured to own the whole of the accounting files and processes. These commands have been designed to be run using cron in batch mode. It is still possible to execute these commands manually from a command line or script.

- ckpacct Controls the size of the `/var/adm/pacct` file. When the `/var/adm/pacct` file grows larger than a specified number of blocks (default = 1000 blocks), it turns off accounting and moves the file off to a location equal to `/var/adm/pacctx` (x is the number of the file). Then ckpacct creates a new `/var/adm/pacct` for statistic storage. When the amount of free space on the filesystem falls below a designated threshold (default = 500 blocks), ckpacct automatically turns off process accounting. Once the free space exceeds the threshold, ckpacct restarts process accounting.
- dodiskDodisk produces disk usage accounting records by using the `diskusg`, `acctdusg`, and `acctdisk` commands. By default, dodisk creates disk accounting records on the special files. These special filenames are usually maintained in `/etc/fstab` or `/etc/filesystems`.

- **monacct** Uses the daily reports created by the commands above to produce monthly summary reports.
- **runacct** Maintains the daily accounting procedures. This command works with the **acctmerge** command to produce the daily summary report files sorted by user name.
- **sa1** System accounting data is collected and maintained in binary format in the file `/var/adm/sa/sa{dd}`, where `{dd}` is the day of the month.
- **sa2** The **sa2** command removes reports from the `../sa/sa{dd}` file that have been there over a week. It is also used to write a daily summary report of system activity to the `../sa/sa{dd}` file.
- **prtacct** Displays total accounting files.
- **sa** Summarizes raw accounting information to help manage large volumes of accounting information.
- **sadc** Reports on various local system actions, such as buffer usage, disk and tape I/O activity, TTY device activity counters, and file access counters.
- **time** Prints real time, user time, and system time required to execute a command.
- **timex** Reports in seconds the elapsed time, user time, and execution time.
- **sar** Writes to standard output the contents of selected cumulative activity counters in the operating system. The **sar** command reports only on local system actions.

System Accounting Commands that can be Run Automatically or Manually

The following system accounting commands may be run either from the command line or in an automated startup script:

- **startup** When added to the `/etc/rc*.d` directories, the startup command initiates startup procedures for the accounting system.
- **shutacct** Collects entries associated with what time accounting has been turned off by calling the **acctwtmp** command to write a line to the **wtmp** file. It then calls the **turnacct off** command to turn off process accounting.

Note

A number of system five UNIX varieties require that the `/etc/rc` files are edited to enable the system accounting run configuration.

Manually Executed Commands

The following commands are designed to be run from the command line and provide the functionality listed in the following table:

- **ac** Prints connect-time records.
- **acctcom** Displays process accounting summaries. (this file may generally be accessed by all users).
- **acctcon1** Displays connect-time summaries.
- **accton** Turns process accounting on and off.
- **chargefee** Charges the user a predetermined fee for units of work performed. The charges are added to the daily report by the **acctmerge** command.
- **fwtmp** Converts files between binary and ASCII formats.
- **last** Displays information about previous logins.
- **lastcomm** Displays information about the last commands that were executed.
- **lastlogin** Displays the time each user last logged in.
- **prctmp** Displays session records.

TCPWrappers – limiting (or adding) access to the Host

TCPwrappers works in conjunction with the `/etc/hosts.allow` and `/etc/hosts.deny` files in order to restrict access to specific network services. The configuration of these services is provided through separate allow or deny statements. Together, this allows for an extremely granular set of access control lists. As an example, the subsequent configurations file denies access to everyone (in `/etc/hosts.deny`) creating, in effect, a default deny rule. In the succeeding `/etc/hosts.allow` configuration file, access to chosen trusted hosts is selectively allowed.

`/etc/hosts.deny`

```
# hosts.deny This file describes the names of the hosts
#           which are
#           *not* allowed to use the local_INET services, as
#           decided
#           by the '/usr/sbin/tcpd' server.
```

ALL: ALL

`/etc/hosts.allow`

```
# hosts.allow This file describes the names of the hosts
#           which are
#           allowed to use the local_INET services, as decided
#           by the '/usr/sbin/tcpd' server.
```

```
# allow access to local machines
#
```

ALL: localhost, .farm.ridges-estate.com

```
# Other trusted systems - anyone from domain at Bagnoo
# except for my when my systems are visiting
#
```

ALL: .guest.ridges-estate.com EXCEPT sister.ridges-estate.com

```
# allow FTP access to anyone inside the Bagnoo Test Networks
#
in.ftpd: .test.ridges-estate.com
```

Several other *NIX network applications and services have the ability to restrict access. The Apache web server can use the “access.conf” to restrict access at the directory-level to hosts and domains. Further, SSH incorporates the ability to restrict access to selective hosts and address ranges in the server configuration file.

ssh is far better than telnet, rlogin, rsh and rcp

Secure Shell (ssh) is a “program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels”. (Fsecure)

SSH should always be used to replace all the first-generation tools, like telnet, rexec, rlogin and rcp. This is even more critical in unsecure environments, such as when used across the Internet, as it is possible that an attacker could be eavesdropping on the network with packet sniffers. Older style protocols, like Telnet, send authentication information and subsequent communications in clear text. Not only is there a problem with packet sniffing, but an attacker could also hijack the session.

SSH provides the capability to offer public key encrypted tunnels that provide protection against packet sniffing and hijacked connections. These tunnels may be used to encapsulate other protocols allowing for the provision of secure X11 sessions and the capability to redirect TCP/IP ports. As such, other TCP/IP traffic may be encrypted through the introduction of an SSH –based tunnel. There are both open source and commercial SSH clients and server software for *NIX. In addition, Windows and Macintosh clients also exist in both the commercial and open source realms.

inetd and xinetd

Network services on UNIX start in a variety of different ways. A common method used by many applications is the “Super Daemon”. A daemon on a UNIX system is a process or service that is initiated and which subsequently continues to run without further interaction. It may initiate further actions from time to time or may wait for a network connection before taking any other action. SMTP (the mail daemon) is an example of such a service. The mail forwarder will bind socket (generally to TCP port 25) and wait for a connection from another mail server before it does anything. The two super daemons are inetd and xinetd. inetd has no access control built into itself by default. It was

the original version of the software. Although both versions may be found on most UNIX systems, the added functionality and increased security of xinetd makes it the better choice.

The configuration of xinetd is not the same as that for inetd. Instead of a separate consideration file (as is used by inetd), xinetd relies on a particular directory (usually /etc/xinetd.d). This directory generally contains an individual confederation file for each of the services that are available and which are set to run at boot on the system. The auditor needs to note that services may be run even without a valid configurations file and, in some instances, services may not be running where there is a valid configurations file. In some instances, services may have a configurations file that is marked `disable = yes!`

The primary reason for choosing xinetd over inetd is that xinetd integrates *TCPWrappers* into xinetd in order to allow access controls for the individual services. This means that access control through ACLs is offered by the “Super Daemon” without a requirement to call tcpd for each of the services as they are launched.

And, in the majority of systems (unless specially configured), inetd services will not have particularly strong authentication methods associated with them. Further, inetd –based services do not generally log individual accesses to syslog. TCPwrappers adds the capability to screen access based on the client’s IP address, creating a simple host-based firewall. This also has the capability to log both successful attempts to access the service and also failed attempts. These logs will contain the IP address of the system that has accessed or attempted to access the service. Configuring the access control lists (ACLs) used by TCPwrappers does potentially take some time and a fair bit of planning. The upside, however, is that once this file is in place and running, the system will be far more secure. One of the key principles of defense in depth is to not rely on single points of failure. Your site may have firewalls at different points on the network, but the addition of access control lists on the system increases the security further for very little cost.

Network profiling

It is essential to identify network services running on a *NIX host as a part of any security test or Penetration test. To do this, the system security tester needs to understand the relationship between active network services and local services running on the host, and be able to identify network behavior that occurs as a result of this interaction. There are a number of tools available for any *NIX system that the system security tester needs to be familiar with.

Netstat

Netstat lists all active connections as well as the ports where processes are listening for connections. The command `netstat -p -a --inet` (or the equivalent on other *NIX'es) will print a listing of this information. Not all *NIX versions support the `netstat -p` option for netstat. In this case, other tools may be used.

Turning Off Services in UNIX

This process will vary depending on the version of UNIX or Linux being run. Most settings are contained within configuration files, though some UNIX systems (such as HP-UX) have a registry system. Always ensure that you have thoroughly investigated the system that you are going to attempt to penetrate prior to beginning any test. Turning off services (such as cron and the log daemons) can make accessing a system easier. It may also act as a flag to show what is occurring.

Controlling Services

In order to understand *NIX services, it is important to also understand how these services are started. To do this, we will take a brief look at how their underlying stack may be configured. The reason for this is that individual services will be impacted through the underlying configurations. The file, `/etc/sysctl.conf` is common to the majority of *NIX systems. The contents, configurations and memory processing will vary across systems. The sysctl (System Control) configuration will, in the majority of cases, control the system configurations that are of prime importance to the auditor. All of the options listed below may not be found in this file, but they may be included in one format or another.

ip_forward

This option lets the IP stack act as a router and forward packets. Multiple interfaces are not required for this functionality.

accept_source_route

This setting configures the operating system to accept source routed packets.

tcp_max_syn_backlog

This setting allows the configurations of the maximum number of SYNs in the wait state.

rp_filter

This setting provides basic IP spoofing protection for incoming packets.

accept_redirects

This setting configures the network stack to accept redirect messages and allow them to alter routing tables.

tcp_syncookies

This setting provides syn-cookie based protection against syn flood DOS attacks.

send_redirects

This setting controls whether or not the system can generate redirect messages.

accept_redirects

This setting is a secondary control used to configure redirect acceptance behavior.

The auditor should create a script to test these settings. The benefits are twofold:

- The settings may be initially tested against an agreed baseline standard, and
- The settings may be tested over time such that a system may be compared to its baseline standard and also a change log.

Enabling .rhosts

It is important to ensure that that no user (not even – and in fact especially not – root) has an “.rhosts” file in their home directory. The “.rhosts” file is one of the biggest security risks and is in fact a greater risk than `/etc/hosts.equiv` file, although they have the same functional purpose. The problem with “.rhosts” files is that they can be created by each user on the system. Some services, such as running unattended backups over a network, try to use these files. However, it should be avoided.

The security tester should attempt to create a `cron` process that will be implemented to alter the contents of any of these files and add a .rhosts file, if possible. Creating any `$HOME/.rhosts` file is a start. The system should be configured with a blank or empty file owned by root that can only be accessed or written by root (that is 400 permissions). The goal is to be able to create a file of this type that has the symbol “-” as the first character in this file, or the symbol “+” on any line, as these may allow users access to the system. Attempt to create files in any user’s home directory.

What is running on the system?**Lsof**

The command, “lsof” allows the system security tester to list all open files where “*an open file may be a regular file, a directory, a block special file, a character special file, an executing text reference, a library, or a stream or network file*”.

LSOF is available from <ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/lsof.tar.gz>.

Ps

The command, “ps” reports a snapshot of the current processes running on *NIX host.

Some examples from the “ps” man page of one UNIX system are listed below. To see every process on the system using standard syntax:

- ps -e
- ps -ef
- ps -eF
- ps -ely

To see every process on the system using BSD syntax:

- ps ax
- ps axu

To print a process tree:

- ps -ejH
- ps axjf

To get info about threads:

- ps -eLf
- ps axms

To get security info:

- ps -eo euser, ruser, suser, fuser, f, comm, label
- ps axZ
- ps -eM

To see every process running as root (real & effective ID) in user format: ps -U root -u root u.

To see every process with a user-defined format:

- ps -eo pid, tid, class, rtprio, ni, pri, psr, pcpu, stat, wchan: 14, comm
- ps axo stat, euid, ruid, tty, tpgid, sess, pgrp, ppid, pid, pcpu, comm
- ps -eopid, tt, user, fname, tmout, f, wchan

Print only the process IDs of syslogd:

- ps -C syslogd -o pid=

Print only the name of PID 42:

- ps -p 42 -o comm=

Top

The command “top” is distributed with many varieties of *NIX. It is also available from <http://www.unixtop.org/>. The top command provides continual reports about the state of the system, including a list of the top CPU using processes. This command gives much of the information found in the Microsoft Windows Task Manager. The main functions of the program as stated by the developers are to:

- provide an accurate snapshot of the system and process state,
- not be one of the top processes itself,
- be as portable as possible.

Kernel Tuning for Security

The *NIX kernel has many configurable parameters that are security related. These parameters can be adjusted to strengthen the security posture of a system covering aspects such as ARP timeouts, IP forwarding of packets, IP source routing of packets, TCP connection queue sizes, and many other factors controlling network connections. Correct tuning of the kernel will even significantly reduce OS fingerprinting of the system when an attacker is using tools as queso and nmap.

Most modern *NIX systems have introduced the concept of the `/proc` file tree. This allows administrators to access the process space and kernel through the file system. The `/proc/<PID>/cwd` may be accessed if you know the identity of a process. Each of the directories in the `/proc` file tree is associated with the PID (process ID) of a running process.

The command “lsdf” (list open files) may be used to identify hidden file space and report on which process is accessing any open file. There are times when it is possible to access a file through the `proc` file system after it has been deleted.

Each variety of *NIX will have its own kernel parameters. It is important that the system security tester investigates these prior to the security test or Penetration test and creates a list of customized settings to check. As an example, we will look at some of the settings in a Solaris *NIX system.

Solaris Kernel Tools

The tool provided within Solaris UNIX for tuning kernel parameters is the command “ndd”. This is far more limited with respect to kernel tuning, as Solaris “ndd” only supports the TCP/IP kernel drivers. This tool is valuable to the system security tester as it can be used not only to set the values of parameters for these drivers, but also to display the current configuration.

Solaris Kernel Parameters

The standard “ndd” command format is:

```
ndd /dev/<driver> <parameter>
```

In the command format, the parameter `<driver>` may be: ARP, IP, TCP, or UDP. The command to view all parameters for a particular driver is:

```
ndd /dev/<driver> \?
```

The command used to set a kernel parameter using `ndd` is (although this is not something that an system security tester will general use):

```
ndd -set /dev/<driver> <parameter> <value>
```

The primary difficulty with Solaris is that any changes to the kernel parameter values using `ndd` are not permanent and will return to default upon system reboot. These changes need to be put into a shell script that is run at system boot to be effective.

ARP

ARP (Address Resolution Protocol) is used to dynamically map layer-3 network addresses to data-link addresses. The ARP cache is vulnerable to ARP cache poisoning and ARP spoofing attacks. ARP cache poisoning involves the insertion of either a non-existent ARP address or an incorrect ARP address into a system’s ARP cache. This results in a denial of service since the target system will send packets to the peer’s IP address but the MAC address will be wrong.

ARP spoofing can be used by an attacker in order to attempt to compromise the system. ARP spoofing relies on disabling a host on the network so that it cannot reply to any ARP request broadcasts and then subsequently configuring the disabled host’s IP address on the attacking host. When the host being attacked attempts to communicate with the disabled host, the attacker’s system responds to any ARP request broadcasts, thus inserting its MAC address in the attacked host’s ARP cache. Communication between the two hosts can then proceed as usual. It is very tricky to protect a system against ARP attacks. A possible defense against ARP attacks is to reduce the lifetime of cache entries. The cache lifetime is determined in Solaris by the kernel parameter `arp_cleanup_interval`. The IP routing table entry lifetime is set by the kernel parameter `ip_ire_flush_interval`. These commands will be set as follows:

```
ndd -set /dev/arp arp_cleanup_interval <time>
nnd -set /dev/ip ip_ire_flush_interval <time>
```

In the `ndd` command, `<time>` is added in milliseconds. Reducing the ARP cache timeout interval and the IP-routing table timeout interval can make it more difficult for the attacker, slowing down their attack. Alternately, static ARP addresses should be created for secure trusted systems. Static ARP cache entries are permanent and therefore do not expire. These entries can be deleted using the command `arp -d`. This may be further enhanced in the event that only static ARP is necessary.

IP Parameters

The Solaris kernel also introduces the capability to modify various characteristics of the IP network protocol. This functionality is provided through the following parameters:

```
ip_forwarding
ip_strict_dst_multihoming
ip_forward_directed_broadcasts
ip_forward_src_routed
```

IP forwarding involves routing IP packets between two interfaces on the same system. Unless the system is action router, IP forwarding should be disabled by setting the kernel parameter `ip_forwarding` to 0 as follows:

```
nnd -set /dev/ip ip_forwarding 0
```

Setting the parameter `ip_strict_dst_multihoming` to 0 lets the system drop any packets that seem to originate from a network attached to another interface, such as a spoofed packet:

```
nnd -set /dev/ip ip_strict_dst_multihoming 0
```

Directed broadcasts are packets that are sent from one system on a foreign network to all systems on another network. Directed broadcasts are the basis for the “smurf” attack, where forged ICMP packets are sent from a host to the broadcast address of a remote network. To disable the forwarding of directed broadcasts, set `ip_forward_directed_broadcasts` to 0 as follows:

```
nnd -set /dev/ip ip_forward_directed_broadcasts 0
```

Source routing is a common attack used to bypass firewalls and other controls. Disallow IP-forwarding to silently drop source-routed packets by setting the Solaris kernel parameter `ip_forward_src_routed` to 0 as follows:


```
ndd -set /dev/ip ip_forward_src_routed 0
```

TCP Parameters

SYN flooding is a common denial of service used against many operating systems. The Solaris kernel commands both provide some protection against SYN flooding and the ability to determine if a Solaris system is under a TCP SYN flood attack by monitoring the number of TCP connections in a SYN_RCVD state as follows:

```
netstat -an -f inet | grep SYN_RCVD | wc -l
```

This is where having a system baseline becomes invaluable as it is then possible to compare the values taken when the machine is running under normal circumstances against those when you believe you are being attacked. Solaris also provides the capability to determine if the system is undergoing a SYN attack using the following command:

```
netstat -s -P tcp
```

The output of this command will provide the tcpTimRetransDrop and tcpListenDrop parameters. An experienced system administrator should be able to recognize a SYN attack using these values. The value tcpTimRetransDrop displays is the number of aborts since boot time, due to abort time expirations. This value includes both the SYN requests as well as established TCP connections. The value tcpListenDrop displays the number of SYN requests that have been refused since the system was booted due to a TCP queue backlog. It is likely that the system is experiencing a SYN attack in the event that the tcpListenDrop value increases quickly along with the value of tcpTimRetransDrop.

It is possible to defend against this type of attack by shortening the value of the abort timer and lengthening the TCP connection queue, both of which may be done through the kernel parameters. To decrease the abort timer, the kernel parameter, tcp_ip_abort_cinterval is used where the value is supplied to the command in milliseconds. The default abort timer interval is 180 seconds. In order to decrease the abort time to 30 seconds, the following command may be used:

```
ndd -set /dev/tcp tcp_ip_abort_cinterval 30000
```

The kernel parameter tcp_conn_req_max_q0 controls the queue size for TCP connections that have not been established. The default value for tcp_conn_req_

max_q0 is set at 1024 queue connections and may be increased using the following command:

```
ndd -set /dev/tcp tcp_conn_req_max_q0 4096
```

Another type of DoS attack involving the SYN flag is based on an attacker exhausting the TCP established connection queue. The TCP connection queue control is given by the kernel parameter tcp_conn_req_max_q which is set by default at 128. An example command to increase the established TCP connection queue would be:

```
ndd -set /dev/tcptcp_conn_req_max_q<size>
```

Security and the cron System

Depending on the version of *NIX, the cron daemon can live in a variety of directories, such as /var/spool/cron or /var/cron. Crontab entries are run at periodic instances by the cron daemon, “crond”. Schedules may be configured to run across a variety of periods based on:

- Month
- Week
- Day
- Hour
- Minute

There are a number of reasons why cron is of particular interest to a system security tester. Firstly, a number of tasks may be automated. The creation of a set of test scripts allows the system security tester to have validation scripts run which send information at preset times. These scripts can be configured to load into a database and validate any changes to the system. Any variation from the baseline or from the previous security test or Penetration test results creates an automated change alerting system and helps to maintain the integrity of the system. System administrators may also use such a system to monitor key attributes, such as memory use and disk capacity. Next, the security of both the crontab itself and the scripts it calls are of paramount concern. If either the crontab process or any of the scripts that it calls are compromised, the entire system is at risk. Many system administrators understand the need to protect the cron daemon but do not understand the need to protect the files that cron calls. When you think about it, however, the matter becomes clear. Cron runs scripts and applications generally as a privileged user. In many cases, this can be as “adm” or even root. If, for instance, an attacker modifies a secure script, they could have run a process to escalate their privileges or even install a root kit. It is not uncommon to see installations where cron files are

calling scripts that have the permission of “777” associated to them. This, in effect, would allow any user on the system to change the script. One of the tasks that anyone testing the security of a *NIX system must do is ensure that all applications and scripts listed in a crontab file are restricted such that only the owner can write to or modify them.

Backups and Archives

It is inevitable that something will eventually go wrong. There is no difference to the statement when you consider *NIX, Windows and it doesn’t matter what operating system you use. Consequently, there needs to be some means of ensuring that the data on the system, and even the system itself, may be recovered. One of the roles of the system security tester is to ensure that processes are in place that will lead to this end. There are a number of ways to ensure that a *NIX system is adequately backed up and archived including both commercial options and those that come with the system. We will discuss only those tools that come with *NIX for the time being.

tar, dump, and dd

The “tar”, “dump” and “dd” commands provide the system security tester with a simple means of collating files to either get them to or from the *NIX system being tested.

tar

The “tar” command is short for tape archiving, the storing of entire file systems onto magnetic tape, which was the origin of the command. However, the command has become a tool to simply combine a few files into a single file allowing for straightforward storage and distribution of backups, archives and even applications.

The process used to combine multiple files (and remember directories are also files in *NIX) into a single file is supplied by the command:

```
"tar -cvf destination_file.tar input_file_1input_file_2"
```

The “f” parameter lets “tar” know that you want to create a tape archive and not to just concatenate a number of files. The “v” parameter places “tar” into verbose mode which reports all files as they are added. The command to split an archive created by tar into separate files at the shell prompt is:

```
tar -xvf file.tar
```

Compressing and uncompressing tar images

Many *NIX varieties use GNU tar, which also allows the use of gzip (the GNU file compression program) in con-

junction with tar to create compressed archives. The command to create a compressed tar archive is:

```
tar -cvzfdestination_file.tar.gz input_file_1 input_file_2
```

The “z” parameter instructs tar to gzip the archive as it is formed.

To unzip a gzipped tar file, the command would be:

```
tar -xvzf file.tar.gz
```

Where a *NIX system does not support GNU tar, gzip may be installed to create a compressed tar file. The following command provides this capability:

```
tar -cvf - input_file_1input_file_2|usr/bin/gzip >
destination_file.tar.gz
```

Alternatively, the Unix compress command may be used instead of gzip. To do this, just replace the “gzip” command with the “compress” command and change the “.gz” extension to “.Z”. Though the extensions do not make a difference to the *NIX system, this is a common designation for the compress command which is set by default to specifically look for an uppercase Z. To divide a “tar” archive that was created and compressed through the use of “gzip”, use the following command:

```
/usr/bin/gunzip -c file.tar.gz | tar -xvf -
```

Likewise, you would divide a tar archive that was compressed using the Unix compress command by replacing “gunzip” with the “uncompress” command.

*NIX does not generally care about extensions in the manner that other operating systems do. However, it is good form to use the right ones so as to not confuse people and run the wrong commands. The extensions “.tgz” and “.tar.gz” are equal to each other and each signifies a tar file zipped with gzip.

dump

The *NIX man pages tell us that “*The dump utility is best suited for use in shell scripts, whereas the elfdump(1) command is recommended for more human-readable output.*”

This command is an effective means of backing up a *NIX system. See the man pages for the version of *NIX being reviewed for details.

dd

The command “dd” is a widespread *NIX command with the primary purpose of providing low-level (actually, bit level)

copying and conversion of raw data. “dd” is an abbreviation for “data definition”. This command is used in digital forensics, as it can create a “byte-exact” copy of a file, drive sector or even an entire drive (even the deleted files).

Some people have termed “dd” with the name “destroy disk” and “delete data” due to its capability to also write an image back to a drive. This provides the capability to both recover a drive (by restoring an image to another disk) or to “wipe” a disk. The wipe process is accomplished by sending either random data (`/dev/random`) or zeros (`/dev/null`) to the drive through “dd”. This is mainly a concern to a system security tester in case of forensic audits for investigations or in ensuring that the system administrators are correctly destroying drives that are destined to leave the organization. Many regulations and standards (HIPAA, PCI-DSS, etc.) require a process to ensure that data has been cleansed. Using “dd” can achieve this.

Logging

There are a wide variety of logging functions and services on UNIX. Some of these, such as the Solaris audit facility, are limited to a particular variety of UNIX. It is important that the tester become familiar with the logging deployed on the UNIX system that they are assessing. In particular, have a look at the syslog configuration file and the `/var/log` and `/var/run` directories and check if there are any remote log servers. Syslog is a network service that is most commonly run locally. This allows for the capability of sharing logs to a remote system.

Know where a system’s logs (if any) are being sent.

Syslog and Other Standard Logs

There are five primary log files that will exist on nearly any UNIX system (the location may vary slightly). These have been listed in the table below.

The 5 primary Unix Log files

- `/var/log/btmp` btmp contains the failed login history
- `/var/log/messages` is the default location for messages from the syslog facility
- `/var/log/secure` is the default log for access and authentication
- `/var/run/utmp` utmp contains summary of currently logged on users
- `/var/log/wtmp` wtmp details the history of logins and logouts on the system

The bad login attempt file (`/var/log/btmp`) is a semi-permanent log (such as `wtmp`) that tracks failed login attempts. This file is a binary format and is read using the

“*lastb*” command. In many systems, the `btmp` file will not be created by default. If this folder does not exist, the system will not log to it. Any test of the security of a *NIX system should validate the existence of this file and ensure that it is functioning correctly – and that it cannot be simply deleted or changed.

A means to validate that this file is working correctly is to attempt to log into the system using a set of invalid credentials. If the log is working correctly, an entry should be recorded noting the failed attempt. It is important that this file is restricted so that only root can access or change it. General users have no reason to see failed attempts and should never be able to change or delete this file. If the tester can change or, worse, delete this file (or if it does not exist), no record of the test will remain.

The messages log (`/var/log/messages`), or at times also the default syslog (on some systems this file will be named `/var/log/syslog`), contains by default the sum of the system messages. Depending on the consideration of the syslog configuration file (commonly `/etc/syslog.conf`), this may contain failed drivers, debug information and many other messages associated with the running of a UNIX system.

The “secure” log (`/var/log/secure`) is designed to record the security and authentication events that occur on the system. By default, applications such as TCPwrappers will log to this file. In addition, the PAM system and “login” facilities will write to this file on most UNIX systems.

The `utmp` file (`/var/run/utmp`) contains a point in time view of the users that logged on to the system. This file is used by a number of applications and utilities (such as the “finger” and “who” commands). This file is volatile in that it will not survive a system boot. Further, when the user logs out of the system, their entry is removed.

This file does not contain historical data. It is possible to gain a snapshot of user information at a point in time through this file. This information includes the username, terminal identifier, the time that the user logged in to the system and also where they log in from (which may be a local TTY or remote network host). Most rootkits will change the functionality of this file in an attempt to hide themselves. This is also the goal of many security testers.

The `wtmp` file (`/var/log/wtmp`) is a binary file similar to “utmp”. This file is also utilized by applications such as “finger”, “last” and “who”, and contains much of the same information as “utmp”. The primary difference, however, is that it is more permanent in nature. This file provides a formal audit trail of user access and will also record system boots and other events. This file is commonly used when investigating an incident. The “last” command uses this file to display a list of accesses to the system. It will

display a historic list as well as listing any user who was still logged onto the system. Like many other UNIX logging facilities it must be activated.

Most UNIX systems (and any that are configured correctly) will rotate logs periodically. This may be done through an automated facility such as “cron” or through some other application. It is important to both verify and validate how the log files are being rotated, whether they are being stored in an offline facility, but they have been backed up, and lastly that they are maintained online for an adequate period of time.

Always consider long-term data retention needs and the capability to restore logs after an extended period of time when testing a system. Some logs are stored as volatile memory and sending large amounts of data to a system will wipe the early logs.

Tricks and Techniques

In the previous section, the common commands in a *NIX system were introduced. It is more important than most people think to be extremely well versed in the use of the standard administrative and user commands on a *NIX system when testing it.

A penetration tester who does not become an expert in the basic commands will never become an expert tester. There are numerous reason for this assertion, not least being that the tester will leave a trail that is simple to find and obvious to detect. Every time you make a mistake on a live system, you increase the probability that you will be caught. On top of this, you increase the amount of data that can be used to follow your tracks.

So, this means that all of the prior commands are important. Many beginning penetration testers believe that it is more important to find some “special tool” or dedicated

In “file linking”

The “ln” command is an important tool in the tester’s arsenal. It is used to either:

- Create a link to a target file with a selected name.
- Create a link to a target file in the current directory.
- Create links to each target in a directory.

The “ln” command will by default produce hard links. Symbolic links are created with the “--symbolic” option set (or “-s”). In order to create a hard link, the target file has to exist. The primary formats of the commands are:

- ln [OPTION]... [-T] <TARGET> <LINK_NAME>
- ln [OPTION]... <TARGET>
- ln [OPTION]... -t <DIRECTORY> <TARGET>

The chief reasons to use this command are in hiding files and creating rootkits or cuckoo programs. The “ln” command need not alter any files on the host. Yet, it can be used to redirect a user or process to another file. It can also be used to replace a file or to create one that is linked to a rogue binary in another location.

It is not only binaries. Configuration files and even devices (/dev/*) can be linked. As an example, many users do not check their path. It is all too common to see users and even administrators run a binary without setting the path to the file. For instance, running “sh” instead of entering /bin/sh. Worse, path and many directories are often poorly configured.

In previous tests, I have noted this issue many times. The following example is from a real financial system.

- The default path for root was /usr/local/bin:\$PATH
- The directory permissions on the /usr/local/bin directory were 775, allowing any user on the system to create a file.
- The shell “/bin/sh” exists, but there is no /usr/local/bin/sh shell or binary.

The answer in this instance (and all of those like this) is to create a cuckoo shell to install other programs. This would be created as /usr/local/bin/sh with the command:

```
“ls -s /tmp/sh /usr/local/bin/sh”
```

Where the file /tmp/sh can be as simple as a script that does a set of commands and returns to the shell such as:

```
#!/bin/sh
/tmp/do.evil.stuff.sh
'/bin/rm /usr/local/bin/sh'
'rm -r /tmp'
/bin/sh
```

The question will arise, “why not just copy a file directly to the link location?” The answer to this is that it will affect forensic analysis of the system later. It is possible to load files into memory space and into memory file systems. It is far more difficult to recover memory file systems then to recover deleted files on disks.

The symbolic link file will be recoverable, but possibly not its contents.

In the example above, as the cuckoo shell is earlier in the path, the linked shell or binary in the /tmp directory will run instead of the desired /bin/sh shell command. This could occur through poor scripting (where the full path is not included in the script header) or

when a “lazy” user or administrator types “sh” rather than “bin/sh”. On top of this, you can redirect configuration files when a service starts with far less impact than changing the file itself.

A final use is to link the “touch” command into a memory location. The “touch” command can be used to change the last modified and accessed times on a file. This can be used to hide the fact that a command was executed or that a file was read. The problem is that the touch command will display a modified access time. This is an indication of its being run (logs may also exist, but these are checked less often).

By creating a “hard link” into the *NIX memory file system, a copy of the command can be created in full. Hence, the touch command can be run and leave only volatile data as evidence of its use.

Netcat (NC)

The netcat command is an example of what you can do with a simple programme that is designed to redirect traffic/Standard Out/, etc.

The obvious use is in moving files. There are a number of less obvious uses as well. Some of these include:

- Port and Protocol Scanning
- Vulnerability Scanning
- Testing and making connections to open ports
- Acting as a virtual server or honeypot
- A simple port-scan logger
- A replay attack engine
- A backdoor into a system and a remote shell
- A forwarder and Relay

I shall be covering some of these uses of this one simple command in the following section of this paper.

The command options for netcat are:

- -d detach from console, stealth mode
- -e prog inbound program to exec
- -g source-routing hop point[s], up to 8 addresses
- -G num source-routing pointer. The options are 4, 8, 12, etc...
- -iN for “N” seconds delay interval for lines sent, ports scanned
- -l listen mode, for inbound connects
- -L listen harder, re-listen on socket close
- -n numeric-only IP addresses, no DNS names – no resolving
- -o file hex dump of traffic
- -p port local port number
- -r randomize local and remote ports

- -s address local source address
- -t answer TELNET negotiation
- -u UDP mode
- -v verbose mode
- -vv be more verbose
- -wN for N secs timeout for connects and final net reads
- -z zero-I/O mode [used for port scanning]

Port Scanning with Netcat

In this section on commands, we are looking at the uses that netcat can provide starting with a less expected use; port scanning. Yes, netcat is not just for connections and sending files between machines. It is also a quick and dirty port scanner.

Netcat will provide the functionality of both a TCP and UDP port scanner. It misses out on many of the refinements of NMAP but, when in a bind, it will do. It will do the standard 3-way TCP handshake as well as collecting UDP responses. Best yet, if you know what you are doing, you can setup Netcat to fire off packets on one system and collect the responses on another. This sounds like another tool...

The options are for both linear sequential scans or there is also an option to randomise the scanning with ports chosen out of sequence. The latter option is used as it is still less likely to be detected than you would expect.

BEST YET

Netcat allows the scan to come from ANY source port. For firewalls that allow (ANY – ANY) from TCP/UDP 53 (DNS) or TCP 80/443 (WWW), this makes testing through a firewall simple.

The command format is as follows:

```
nc -v -p 53 -w4 -z [Target IP address] [Starting port for the scan] - [End port for the Scan]
```

The options:

- -v This tells netcat to provide verbose output. In this mode it will print the details of each connection that is successfully created (i.e. the 3-way handshake completes).
- -w4 “w” is the wait time. In this case, 4 seconds.
- -z This tells Netcat to send only minimal data for a TCP handshake and nothing else. This is known as Zero I/O mode.
- -p Is the Source port that is used. In this case we are scanning from port 53 as if we were a DNS server...
- -r This is the option to randomise the scan.

The command above used TCP. Having a scan of UDP ports is just as simple, all we need do is add the “-u” option to select a UDP scan. A UDP based scan from UDP 53 to scan ports 1 to 1023 of address 192.168.1.100 would use the following format:

```
nc -v -p 53 -w4 -z 192.168.1.100 1 - 1023
```

On top of this, all the standard netcat options and ports are also available. We can port the output to a file (using > or >>) or pipe the output to another programme (using the pipe: “”).

Piping and execution

If a port is detected, there are two options that are readily available. Piping was mentioned above, where we send data to another programme. Next is selected execution. Using the “-e” option of netcat, it will execute a programme on a successful connect.

This is handy if you are testing a particular port’s vulnerability. More on this will have to wait for another post. An example of this would be when a connection is made to a TCP port, in the following command we run a DNS test script:

```
nc -v -p 53 -w4 -z -e /usr/bin/dns-test.sh 192.168.1.100 53
```

Each time in the previous example netcat connects to TCP 53 it will test the system using a script /usr/bin/dns-test.sh. This can be any programme you choose.

Wait you say, this is testing a single host and is not too useful...

This is where the wonders of scripting come into play. Say we want to scan hosts 192.168.10.1 to 192.168.10.254 for web ports (TCP 80); we can use a simple script such as the following.

```
(for f in $(seq 1 254); do nc -v -w3 -z "192.168.10.$f" 80; done)
```

This simple script runs through each of the hosts. With a little thought, you can start to make some quite elaborate scans with just a simple tool.

CRAIG S. WRIGHT

CraigSWright@acm.org

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

? WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BDSP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

✓ WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

i WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

How to Install a Web Server

The goal of this article is to walk the reader through installing and configuring Apache and PHP.

Apache is a web server, meaning that it communicates with other software (usually web browsers) by using the HTTP protocol. This is a human-readable text-based protocol, which is a useful feature in case manual debugging or network inspection is needed. A typical HTTP request (made from the browser to the server) looks like this:

```
GET / HTTP/1.1
Host: ivoras.net
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux i686)
        AppleWebKit/537.36 (KHTML, like Gecko)
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,hr;q=0.6,en-GB;q=0.4,sr;q=0.2,bs;q=0.2
Cookie: foo=bar
If-None-Match: "43ee64-1dcb-4f72ae6666540"
If-Modified-Since: Wed, 16 Apr 2014 15:52:13 GMT
```

And a typical response begins with headers such as these:

```
HTTP/1.1 200 OK
Date: Wed, 24 Sep 2014 08:08:17 GMT
Server: Apache/2.2.29 (FreeBSD) mod_fcgid/2.3.6 mod_ssl/2.2.29 OpenSSL/0.9.8za-freebsd
Last-Modified: Wed, 16 Apr 2014 15:52:13 GMT
ETag: "43ee64-1dcb-4f72ae6666540"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2952
```

```
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

This is informative, for the cases when you encounter raw HTTP network dumps, but the goal of this article is not to analyse the protocol.

After receiving a request, the web server needs to decide how it will be fulfilled. The simplest case is if the request matches a file accessible to the server – it will simply read the file and send it in the response.

A more complicated case is if it turns out that the request references dynamic content in other words, a web application. Here, there are multiple possible additional protocols and options with which the web server communicates with an application server and passes whatever is returned to the browser.

PHP is one such possible application server. It was developed fairly early and because of this, maintains a philosophy that it is only a “filter” which processes input files possibly containing embedded codes. The typical application starting point would be a file named `index.php` which can contain either PHP code only, or some HTML code interspersed with PHP code (though the latter practice is nowadays obsolete and discouraged). In any case, whatever result buffer is built by PHP, it is passed back to the web server.

There are multiple ways in which PHP can be connected to the web server. In the past, the most popular way was “`mod_php`”, in which PHP was built as a module (a library) for Apache and loaded with Apache itself. In this scenario, the PHP interpreter was literally embedded within the web server’s process.

This has proven to have significant downsides. It is wasteful to have the entire PHP interpreter present in memory and ready for all requests (most of which will be

for static files such as images, CSS and JavaScript code). From a security viewpoint, having the large and complex PHP interpreter which may execute unsafe code embedded in the web server may destabilize it or enable security breaches.

The current best practice is to have PHP running in separate processes, usually communicating with Apache via the FastCGI protocol. Again, there are several different ways to accomplish this, but for simplicity this tutorial will use the “mod_fcgid” Apache module which is a FastCGI manager primarily written with PHP in mind.

Installing Apache

We will use the pkg package manager to install the software under discussion. To demonstrate the basic principles of using pkg, it is common to start the process by searching for available software:

```
> pkg search apache
```

This will result in the output of a large number of packages containing the string “apache” in its names, among which there will be these:

```
...
apache22-2.2.29
apache22-event-mpm-2.2.29
apache22-itk-mpm-2.2.29
apache22-peruser-mpm-2.2.29
apache22-worker-mpm-2.2.29
apache24-2.4.10_1
...
```

The “apache22” packages are for the previous version of Apache, while the single apache24 package is for the current version, 2.4, which we will use in this tutorial.

The “mpm” (multi-processing-module) in the package names refers to the way multiple simultaneous requests will be handled (and there is always simultaneous request processing going on). By default, Apache will spawn completely separate, single-threaded processes to handle incoming requests. This is called the “prefork” model and it is wasteful because it takes up memory and other operating system resources, but is the historical and most compatible behaviour. It is almost never used today. The “worker” model will start several server processes, but each of them will also start several internal process threads (25 by default). This drastically cuts down on the number of processes and is more efficient. It is also the most common model used in Apache 2.2. In Apache 2.4, the “event” model is more mature and the

recommended one for general use. In this model, there are separate processes with multiple threads just like in the “worker” model, but the majority of requests are not handled by the individual threads but by a very efficient “event loop” which can either fulfill them directly and with greater concurrency, or off-load them to threads if needed for some reason. Apache 2.4 supports all these models in the same package.

The additional “peruser” and “itk” models are special cases used for bulk hosting. They are similar to the prefork model, but allow for the individual Apache processes to be executed under the credentials of different system users, which is important for maintaining server security.

The installation step needs to be performed as the “root” user, and you can use the “su” command to switch to it.

Apache is installed simply by issuing the “pkg install apache24” command:

```
# pkg install apache24
```

Updating FreeBSD repository catalogue...

FreeBSD repository is up-to-date.

All repositories are up-to-date.

The following 2 packages will be affected (of 0 checked):

New packages to be INSTALLED:

```
apache24: 2.4.10_1
```

```
db5: 5.3.28_2
```

The process will require 71 MB more space.

16 MB to be downloaded.

Proceed with this action? [y/N]: y

```
Fetching apache24-2.4.10_1.txz: 100%4 MB 1.2M/s 00:03
```

```
Fetching db5-5.3.28_2.txz: 100% 12 MB 2.5M/s 00:05
```

```
Checking integrity... done (0 conflicting)
```

```
[1/2] Installing db5-5.3.28_2: 100%
```

```
==> Creating users and/or groups.
```

```
Creating new group 'www'.
```

```
Creating new user 'www'.
```

```
[2/2] Installing apache24-2.4.10_1: 100%
```

```
/usr/local/share/examples/apache24/httpd.conf -> /usr/
local/etc/apache24/httpd.conf
```

The pkg tool may ask you to update the package repository database or the tool itself before installing any new packages, and you should allow it.

Note that the installation process will create a system user called “www”, a system group with the same name, and a configuration file located in /usr/local/etc/apache24. In FreeBSD, the configuration files of the base system are

located in `/etc`, while the configuration files of additional software installed from the packages are located in `/usr/local/etc`. This is because the `/usr/local` directory is in itself a kind of root directory for all software not belonging to the base system.

Configuring Apache

Before Apache can be used, some initial configuration is recommended. The main Apache configuration file is called `httpd.conf`, and it includes other files if necessary. In `httpd.conf`, you should find and edit the line which configures the server administrator's e-mail address (somewhere around line 188 in the file) and change it to your address, similar to this:

```
ServerAdmin ivoras@gmail.com
```

Near the end of the file, you should uncomment the lines including the MPM-specific configuration, and various default settings configuration:

```
Include etc/apache24/extra/httpd-mpm.conf
Include etc/apache24/extra/httpd-default.conf
```

Note that the paths of the files above are relative to `/usr/local`. Next, open the `httpd-mpm.conf` file and for now simply note the various settings available for the `mpm_event_module` case (around line 60).

Configuring a simple web site

It is a best practice to put all significant services into a single, more manageable directory hierarchy. For this workshop, all services will be put in `/srv`. The web applications and similar services interacting with the web server will be put in `/srv/www`. Our example web site will be put in `/srv/www/default`. All these directories need to be created manually, for example like this:

```
# mkdir /srv
# mkdir /srv/www
# mkdir /srv/www/default
```

For convenience, you should change the ownership of the `/srv/www/default` directory to your own user, so that you don't have to switch to the root user every time you need to change something:

```
# chown ivoras /srv/www/default
```

Next, create a new file in the `/usr/local/etc/apache24/Includes` directory, named "default.conf". This file will con-

tain a "virtual host" configuration for our site. Virtual hosts exist to enable a single web server to host several applications or services under different domain names. In production, this means a single server may host e.g. "www.example.com" and "cloud.example.com" addresses.

The "default.conf" file should contain lines like these:

```
<VirtualHost *:80>
ServerName www.example.com
ServerAlias localhost
ErrorLog "/var/log/httpd-default-error_log"
CustomLog "/var/log/httpd-default-access_log" combined
ServerAdmin ivoras@gmail.com
DocumentRoot "/srv/www/default"
<Directory "/srv/www/default">
Options FollowSymLinks
AllowOverride None
Require all granted
</Directory>
</VirtualHost>
```

This is a basic virtual host configuration which serves static files from `/srv/www/default` on the default HTTP port 80. The `ServerName` and `ServerAlias` directives are very important! They control to what addresses this configuration will apply. In the above case, if you try to access the server using a server name which is neither `www.example.com` nor `localhost`, Apache will fulfill the request from default configuration settings (specified in `httpd.conf` by the `DocumentRoot` directive somewhere at line 220), which is not what is wanted here (and in fact you should change the default configuration to point to your default site also).

Apache works by having configuration directives such as `DirectoryRoot` which configure the paths from which the content will be served, and then separate blocks like the `<Directory>` block above which control the security aspects of certain paths.

This `<Directory>` block specifies in the `Options` directive that the web server will follow symbolic links when it searches for files to serve. It specifies that the directories with those files will not be permitted to contain special `.htaccess` files with configuration overrides with the `AllowOverride` directive, and that when security is considered by the `Require` directive, all requests will simply be granted.

You can create a file named `index.html` in the `/srv/www/default` directory containing the following content:

```
<html>
<head><title>Hello, world!</title></head>
<body>
```

```
<h1>Welcome!</h1>
<p>This is my very own web server.</p>
</body>
</html>
```

This is just an example index.html file, you can create your own if you want to.

Starting Apache

The first step before Apache can be started is to enable it to the system in the `/etc/rc.conf` configuration file. In FreeBSD, this file contains almost all of the important system configuration aspects. This includes configuring the hostname, the IP addresses of the network interfaces, and which services can be started.

You need to add the following line to the `/etc/rc.conf` file to enable Apache:

```
apache24 enable="YES"
```

After this line has been added, you can start Apache by issuing the following command:

```
# service apache24 start
```

If everything goes well, you can immediately open a web browser and point it to the address such as *http://www.example.com* (note that it must match one of the `ServerName` or `ServerAlias` directives in your virtual host configuration) and view the `example.index.html` file.

Installing PHP

If you issue a “`pkg search php5`” command, you will be presented with a very long list of packages containing various versions of PHP and its modules. However, the simplest way is to install the default version of PHP by issuing a command such as:

```
# pkg install php5
```

As of the time of this writing, the default version of PHP is 5.4.

Next, the “mod_fcgid” Apache module needs to be installed by issuing the following command:

```
# pkg install ap24-mod fcgid
```

This module needs to be configured in `httpd.conf`, by adding the following line in the section where all the other `LoadModule` directives are located:

```
LoadModule fcgid module libexec/apache24/mod_fcgid.so
```

PHP has to be enabled in individual virtual hosts, in individual `<Directory>` blocks. To enable PHP, you need to change the `<Directory>` block in your default.conf file to match the following:

```
<Directory "/srv/www/default">
Options ExecCGI FollowSymLinks
AddHandler fcgid-script php
FCGIWrapper /usr/local/bin/php-cgi .php
DirectoryIndex index.php
AllowOverride None
Require all granted
</Directory>
```

Note that the Options directive has been changed to enable ExecCGI, that a new script handler has been registered with the AddHandler directive, and that mod_fcgid was enabled by pointing the FCGIWrapper to the php-cgi executable file and telling it to process files ending in .php. Finally, Apache is told to look for files named index.php if the client (the browser) did not specify a filename in its request.

To test this, remove the `index.html` file created previously and create a new file named `index.php`, containing the following lines:

```
<?php
phpinfo();
```

<div> <div>PHP Version 5.4.31</div> <div>  </div> </div>	
System	FreeBSD 10.0-RELEASE-p2 FreeBSD 10.0-RELEASE-p2 #0: Tue Apr 29 17:00:21 UTC 2014 root@amd64-builder.daemonology.net:/usr/src/sys/amd64/amd64
Build Date	Aug 5 2014 15:58:44
Configure Command	'./configure' '--with-layout=OS' '--localstatedir=/usr' '--with-config-file-scan-dir=/usr/local/etc/php' '--disable-all' '--enable-bcmath' '--enable-coverage' '--with-bcmath-dir=/usr/local' '--with-core-libs=/usr/local' '--with-ffi-dir=/usr' '--program-prefix=' '--enable-fpm' '--with-fpm-user=www' '--with-fpm-group=www' '--with-regex-pcre' '--with-bd-libs=CALL' '--prefix=/usr/local' '--mandir=/usr/local/man' '--infodir=/usr/local/info' '--build=amd64-portable-freebsd10.0'
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php
Additional .ini files parsed	/usr/local/etc/php/pear.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	20100625

Figure 1. An example output of the `phpinfo()` function

This simple PHP script will dump the PHP configuration to the browser, which is nice for debugging, but is a serious security risk otherwise.

The new configuration will become active when you restart Apache:

```
# service apache24 restart
```

If you point your browser to the same address as before, you will see a table with PHP's configuration options.

What is SSL / TLS and what is its role?

TLS (transport layer security) is the name given to the protocols which are a continuation (or version upgrades) of the SSL (secure sockets layer) protocols. The most noticeable aspect of these protocols is that they are usually used as authenticated, meaning that there are somewhat strong guarantees about the identity of at least one party in a two-way communication. Another less visible but still vitally important feature of TLS is that it is a complex framework supporting a huge number of options, encryption methods and smaller cryptographic protocols, which can interact with each other, and whose interactions can sometimes be detrimental.

TLS depends on several types of public key cryptography, in which a part of the information used in encryption is publicly and freely shared, and a part of it is private, accessible only to the software performing the actual cryptography. There is somewhat complex math which links these two parts of the information to ensure that data encrypted with one part can only be decrypted or verified by the other, which can be extrapolated into a guarantee that a possessor of the private part is who he appears to be, based on the information in the public part. In the context of TLS, the two parts are the “certificate,” which contains the public key along with a description of whom the key belongs to, and the private key. The initial step in TLS communication involves the exchange and / or verification of the public keys by both parties in the communication channel, which ensures that both know who they are talking to, and which establishes the environment for future secure communication.

After the public key cryptography step in communication setup, the protocol switches to the more efficient symmetrical key cryptography, which simply has a single key used to both encrypt and decrypt the data. Additionally, there is a mechanism for the verification of data integrity, which usually uses a hash function.

Both of the steps can have a number of additional options or variants, which are combined into so-called “cipher suites.” A cipher suite describes which algorithms are

used, and sometimes how they are used. An example of a modern cipher suite is “ECDH+aRSA+AESGCM” which means that it:

- Uses the Ephemeral Elliptic Curve Diffie-Hellman algorithm for the initial setup of a communication channel which cannot be eavesdropped on, and which is also resistant to future analysis because the random keys are discarded after they are used (this is the so-called “perfect forward secrecy” or PFS option).
- Uses the RSA algorithm for proving to each other the identities of the client and the server. In common TLS usage, usually only the proof of the server's identity is considered important, but there is always the option of proving both.
- Uses the AES algorithm for the symmetrical encryption part.
- Uses the GCM (Galois-Counter Mode) variant of the AES algorithm, which allows encryption of individual bytes instead of blocks and which also verifies the integrity of the transmitted data, while also being very efficient.

As can be seen from this, TLS is a complex beast. The cipher suite described above is the current state of the art and is the recommended one for modern applications, but a new set of algorithms is under development and a likely future state of the art cipher suite may be ECDHE-RSA-CHACHA20-POLY1305-SHA256.

Certificate authorities

The name “Certificate authority” (CA) has a technical and a legal aspect. In technical terms, it is the software which generates new certificates from submitted certificate requests. The certificate requests contain information such as the host name (e.g. the virtual host name of a web server), the name, the location, and the owner (e.g. the name of the company) of the service which requires the certificate, and the CA can copy some or all of this information into the certificate before signing it with its own private key. In this way, the CA guarantees that the data in the certificate is correct to the best of its knowledge – which is the legal aspect of its function. The original idea was for the CAs to individually verify all the possible information in the certificates, which is both expensive and takes a long time. To allow cheaper certificates, multiple CA vendors have introduced (often incompatible) “classes” of certificates, in which only a subset of the information is verified. The cheapest classes usually only verify that the server host name for which the certificate is requested is owned by the person or the company requesting the certificate

and nothing more – yet to most of the visitors this appears to be as secure as certificates for which all the company information is also verified. Today, after many breaches of security in known CAs, and the disillusionment in the concept of what CAs actually verify in the certificate data. In almost all cases, the cheapest SSL/TLS certificates are good enough. What is missing in the above description is how exactly a browser verifies that the certificate information it received from the server is trusted. To do this, browsers (and all other software working with TLS) have a list of

trusted “root” CA certificates, and they perform public key cryptography verification to ensure that one of those trusted CAs has signed the certificate it has received from the server (and also that the server has access to the private key which corresponds to it). This is the reason why home-made certificates (also called “self-signed”) make the web sites they are used in appear as “untrusted” to the visitors. New CAs are responsible for sending their own certificates to all the popular browser vendors, for inclusion in the next versions of the browsers (Figure 2).

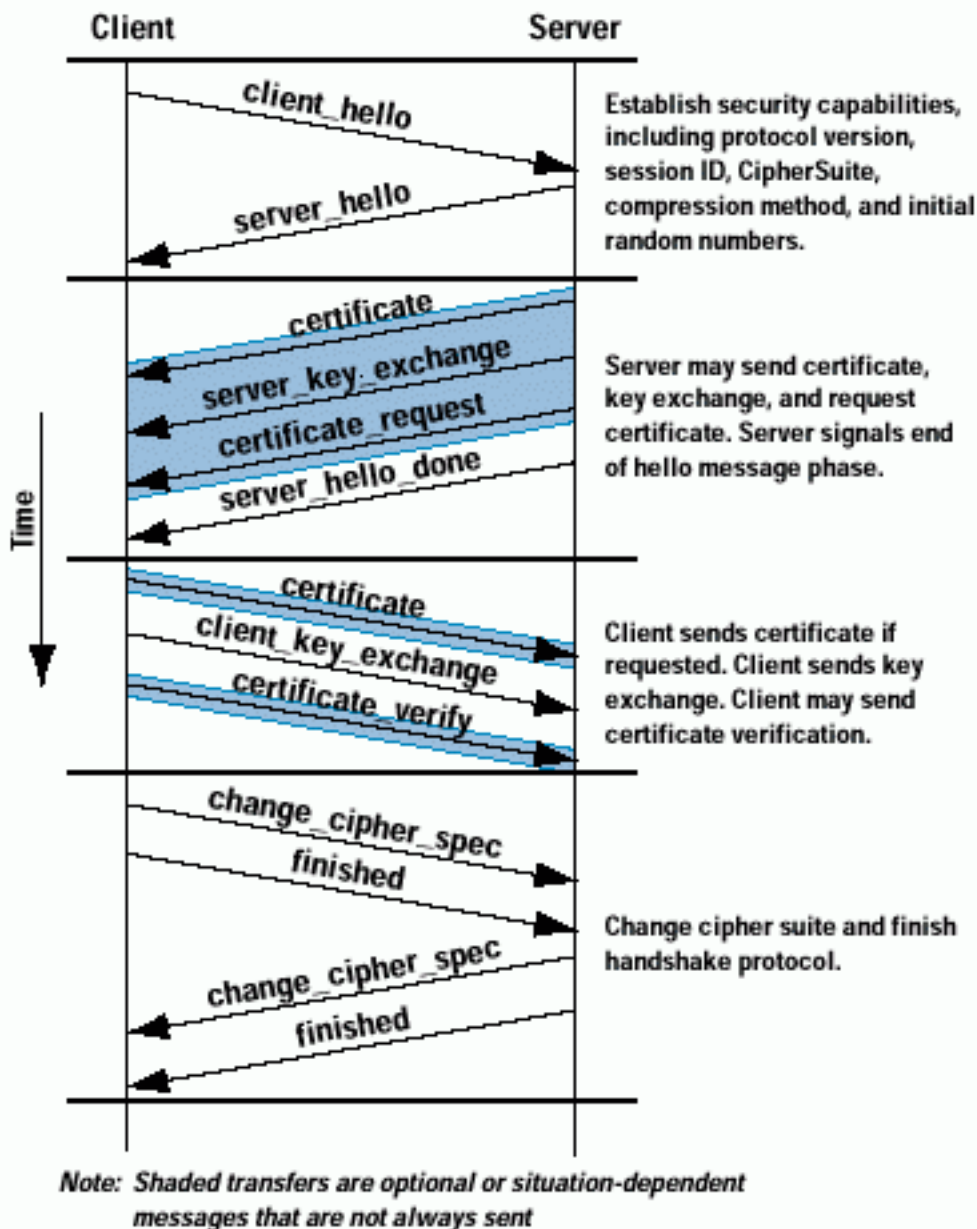


Figure 2. An overview of how a secure connection is negotiated between the client and the server using TLS

Certificates in themselves have few important tunable parameters. The most significant ones are the RSA key size (the “strength” of the public / private keys) and the hash algorithm used for signing and protecting its information. It is very important today (because browsers will reject certificates they do not consider to be secure) to issue certificates with key sizes of at least 2048 bits, and which use the SHA-2 family of hash algorithms (SHA-256 is one of the algorithms in this family).

Certificate requests

Certificate requests are generated together with the private and public keys, and contain the information which is requested to be included in the certificate.

The certificate request and the corresponding keys can be generated with a single OpenSSL command, for example this one:

```
> openssl req -new -newkey rsa:2048 -nodes -sha256 -out
  ivoras.net.sha256.csr -keyout ivoras.net.key -subj "/
  C=HR/ST=Croatia/L=Zagreb/O=Ivan Voras/CN=www.ivoras.net"
```

The above command will generate a new 2048-bit RSA key and save it, unencrypted to the file `ivoras.net.key`. It will generate a certificate request which uses SHA-256 for hashing, and will write it to the file named `voras.net.sha256.csr`. Both files will by default be encoded in the so-called PEM format. The string in the “-subj” parameter contains the attributes identifying the owner of the certificate and the hostname for which the certificate will be valid. In the above example, the following information is present:

- C – The Country of the owner of the certificate (HR = Croatia)
- ST – The State within the country (if applicable)
- L – Location, city or town of the owner of the certificate
- O – Organization name of the owner of the certificate
- CN – The hostname for which the certificate is valid

It is possible to have multiple hostnames in the certificate, either by including multiple CN attributes, by including “alternate names” (which requires using a modified `openssl.cnf` file), or by using so-called “wildcards” in the CN. The latter is a very useful practice since it enables that a single certificate be used on a single host which serves multiple services in different subdomains. For example, a certificate with “CN=*.ivoras.net” can cover “www.ivoras.net”, “mail.ivoras.net” and all other subdomains. Note that the asterisk only stands for a single subdomain level, and that it syntactically doesn’t cover the top-level domain (i.e. *.ivoras.net does not cover `ivoras.net`).

Those interested in creating their own CA should look at the XCA Open source project.

Submitting the certificate request to CAcert.org

In a “real” production environment, you should buy a certificate from a CA whose certificates are included and trusted by the most web browsers, but for testing, you can use the free certificate provider named CAcert.org. It is an open (and free) international project which aims to provide free certificates, whose certificates are supposed to be trusted by simple mechanisms and the mutual trust of its members.

However, they have not yet succeeded in getting their CA root certificates into significant mainstream browsers or operating systems.

The first step after signing up at www.cacert.org is to register the domain(s) you wish to have certificate(s) issued for, by clicking Add on the “+Domains” menu on the right-hand side of the page. CAcert will verify you own the domain by sending a confirmation e-mail to an address which is configured in the DNS system as the owner of the domain, or a similar simple mechanism. Such mechanisms are also used by other, commercial CAs.

After this, you can add new Server Certificates, by pasting the contents of the generated `.csr` file into the web browser. A certificate request as a PEM file looks like this:

```
--BEGIN CERTIFICATE REQUEST--
MIICozCCAYsCAQAwXjELMAkGA1UEBhMCsFIxEDAQBgNVBAgMB0NyYb-
  2F0aWEExDzAN
BgNVBAcMB1phZ3JlYjETMBEGA1UECgwKSXZhbWw3JhczEXM-
  BUGA1UEAwOd3d3
Lml2b3Jhcy5uZXQwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAo-
  IBAQDG9Omh
nRjqCBfw4VLFUSopy6jUIF1UMqEs69ctKbH4GOzjKo3ZOLHsk/
  mYb4LaWBmwK8SV
3QSQGP6HS34mc1N3625eQVMx9ojeYXNDAjBRBGURVv6mckR0FnQELK-
  WoCAkiZ1V2
FdMV9B02neb2g3/3uFUyR5sCznC/juFhdG5Db4md1N5JZRZo0WkbK+mv+
  W5opSG7
mrUyIijxOHTFLTmyO9J8fiFKo7hW0ihJGnH5fITXM/65cEUHpmJn1OHw6
  ufq9Ze
RmtqnqCCTyKhtT+Lu+DP7NgjeE/5u5UVAN8gAS04YpuVff5vdBivZIRB2
  oaVH32r
Eupa2mZ+8EVKK03pAgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAE16UK
  m8LTukx
nc43/1UFiVr5Kq9dkMrqlfAIaihrwbK6a11+sKz86jWHZ2EsqRPyyois
  CdNebw0
dFOdr30fk7iNZbqjmrUqH89KZwnX/2o/5rCdwehf/
  1hdYE+nc4uVgNbx4EwaqEKZ
vTvIjYQ8T1+OcN/WEY22cdio29psvDaEccE4R6xrQWgACxgun93VS-
```

```
m2yOldFu8A5
K0uwyddfNbjyv40zA7JXU36WLYh0hXCuGibX0fIFkddE+eT0h3bKnO2wD
oIz3jWW
syPrYK/a/7XMQfKZdsnTiq1t2fQC/jqn6+6bV4YA5WKymH4idaPBrALSx
fw05ZdA
D1bUjtmh1g==
--END CERTIFICATE REQUEST--
```

After CAcert.org processes the request, the end result should be a certificate which you can install into a web browser. You should put both the certificate and the private key file into a special directory, for example into `/var/ssl`.

Configuring Apache for an HTTPS site

To configure Apache for HTTPS you will need a private key file and a certificate file. In this example, we will adapt the virtual host configuration from the previous tutorial to include the proper configuration for an HTTPS site. In the past, using virtual hosts with SSL/TLS was a problem as the SSL/TLS negotiation happened before the information about the requested virtual host was passed from the browser to the web server. This was addressed approximately 10 years ago with the SNI (Server Name Indication) extension to TLS, and it is now supported by all major browsers.

The first step is modifying the `httpd.conf` file in `/usr/local/etc/apache24` by uncommenting the line loading the SSL module for Apache:

```
LoadModule ssl_module libexec/apache24/mod_ssl.so
```

To start, you should copy & paste the old definition of the virtual host below the original one, add the `Listen 443` directive, and then add several changes to the virtual host so that it looks like this:

```
Listen 443
<VirtualHost *:443>
ServerAdmin ivoras@gmail.com
ServerName www.example.com
ServerAlias localhost
ServerAlias web7.fer.hr
ErrorLog "/var/log/http-default-error_log"
CustomLog "/var/log/http-default-access_log" combined

DocumentRoot "/srv/www/default"
<Directory "/srv/www/default">
Options ExecCGI FollowSymLinks

AddHandler fcgid-script php
FCGIWrapper /usr/local/bin/php-cgi .php
DirectoryIndex index.php
```

```
AllowOverride None
Require all granted
</Directory>

SSLEngine on
    SSLCipherSuite !ADH:!EXPORT:!SSLv2:EECDH+aRSA+AESGCM:EE
    CDH+aRSA+RC4:RC4+RSA:+HIGH:+MEDIUM:+LOW
    SSLHonorCipherOrder On
    SSLCertificateFile /var/ssl/ivoras.net.crt
    SSLCertificateKeyFile /var/ssl/ivoras.net.key

</VirtualHost>
```

Note that the first line was changed to declare this virtual host on the port 443, and the end of the definition contains 4 directives beginning with `SSL` which enable TLS, configure the list of cipher suites and specify the certificate and the key which will be used.

Note that the list of cipher suites is a single line in which parts are separated by the colon (:). This particular line will first try to negotiate `EECDH` with `AES-GCM`, and then fall back to less secure (but still very good) options.

After the configuration file is changed, you need to restart the browser with the command:

```
# service apache24 restart
```

Conclusions

You can now point your browser to the HTTPS address of your site. If you have a certificate which is not trusted by your browser, you will still need to accept it as a security list, but other than that, the configuration should work out of the box.

IVAN VORAS

Ivan Voras is a FreeBSD developer and a long-time user, starting with FreeBSD 4.3 and throughout all the versions since. In real life he is a researcher, system administrator and a developer, as opportunity presents itself, with a wide range of experience from hardware hacking to cloud computing. He is currently employed at the University of Zagreb Faculty of Electrical Engineering and Computing and lives in Zagreb, Croatia. You can follow him on his blog in English at <http://ivoras.net/blog> or in Croatian at <http://hrblog.ivoras.net/>, as well as Google+ at <https://plus.google.com/+IvanVoras>.

Working with Core Dumps in GDB

A core dump or core file is produced when a signal (man signal) indicates that it is not possible to continue executing the program because of an error (for example SIGSEV).

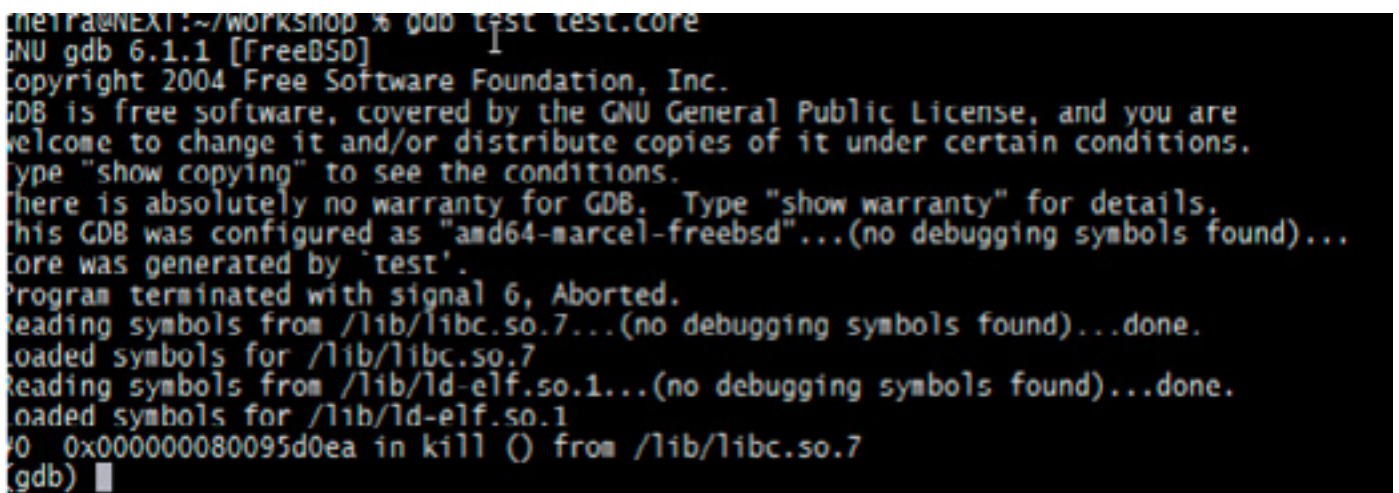
A core file contains the current state of the program when it was executing (the stack of each thread, contents of the cpu registers, the values of global and static variables, etc.). If we are lucky and the program that dumped core was compiled with debug options, the core file will have information about the source code and lines.

A core file is useful to do a postmortem analysis of what happened to our application. If we have the same versions of libraries between hosts, you could take a core produced, let's say in a production machine, and analyze it in another machine that has the same libraries and versions (it is a must or your core file analysis will be invalid).

Allowing to dump cores

For the examples, we are going to use FreeBSD 10.0-STABLE. To check if your installation is enabled to generate core files, you could execute the following commands as root.

```
# sysctl -a |grep kern.coredump
kern.coredump: 0
```



```
netra@NEXT: ~/WORKSHOP % gdb test test.core
GNU gdb 6.1.1 [FreeBSD]
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "amd64-marcel-freebsd"...(no debugging symbols found)...
Core was generated by `test'.
Program terminated with signal 6, Aborted.
Reading symbols from /lib/libc.so.7...(no debugging symbols found)...done.
Loaded symbols for /lib/libc.so.7
Reading symbols from /lib/ld-elf.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-elf.so.1
0 0x0000000080095d0ea in kill () from /lib/libc.so.7
(gdb) █
```

Figure 1. How to see the conditions


```
# sysctl kern.coredump=1
kern.coredump: 0 -> 1
# sysctl -a | grep kern.coredump
kern.coredump: 1
```

`kern.coredump = 0` means that core files will not be generated, just use the `sysctl` (man `sysctl`) facilities to change that, and make it permanent by editing `/etc/sysctl.conf` (man `sysctl.conf`).

You can see the difference by executing this simple program:

```
#include<stdio.h>
#include<stdio.h>
int main() { abort();}
```

Just name it 'test.c' or whatever name you want and do a make test (or make <the name you have chosen>) and you will see it core dumping and leaving a test.core file or none if you did not apply the changes to the `kern.coredump` variable. `Abort()` causes the program to abnormally terminate hence generating a core file.

Working with the core file

So we now have a core file. To take a look at it and start debugging, we just type:

```
% gdb test test.core
% gdb <name of the executable must be in your path> <core file>
```

As you see, there are no debugging symbols. As I used the make command, it took my CFLAGS from the `/etc/make.conf` file and there I did not have the `-g` flag set.

Debugging without the source code

Debugging without the source code could be complicated. If debugging when you have the source code is sometimes cumbersome, let's try it without it!

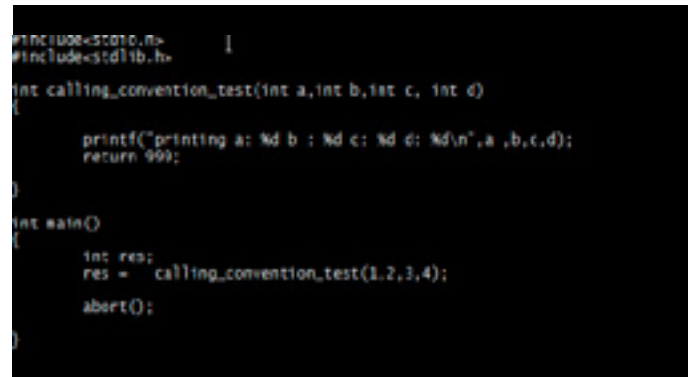
We need to introduce the concept of Application Binary Interface. An ABI defines how system calls parameters are passed (which registers contain which argument), how functions are called and in which binary format information should be passed from one program component to another. FreeBSD ABI conforms to these specifications:

- <http://www.x86-64.org/documentation/abi.pdf>
- <http://www.sco.com/developers/devspecs/abi386-4.pdf>

Both documents are a good read. You could also take a look at the developer's handbook: <http://www.nl.freebsd.org/doc/en/books/developers-handbook/x86-system-calls.html>.

Now maybe we are in a better position to debug some core files.

Let's try this simple program:



```
#include<stdio.h>
#include<stdlib.h>

int calling_convention_test(int a,int b,int c, int d)
{
    printf("printing a: %d b: %d c: %d d: %d\n",a ,b,c,d);
    return 999;
}

int main()
{
    int res;
    res = calling_convention_test(1,2,3,4);
    abort();
}
```

Figure 2. How to start simple program

I'm on x86_64 architecture, so let's see page 18, section 3.2.3: Parameter passing. In a nutshell, it says that integer parameters (CLASS TYPE INTEGERS) will be passed using registers `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8` and `%r9`. Let's check that. Compile the program by typing:

```
$ cc <source.c> -o <exec name>
```

Then run the executable program. That will get you a core file and load this into GDB as usual.

```
$ gdb <exec> <core file>
```

Now we will use the disassemble command (disas is the short form). This command dumps a range of memory as machine instructions. The default memory range is the function surrounding the program counter of the selected frame. Two arguments are interpreted as a range of memory to dump. By default GDB uses the AT&T notation (http://en.wikipedia.org/wiki/X86_assembly_language).

If we want to disassemble a specific function just type:

```
(gdb) disas <function name>
```

But wait we will always have a main function. What about the names of the other ones? You could type

```
(gdb) info func
```

And you will get all function names. This works if the executable has not been stripped (man 1 strip; Figure 3 and Figure 4).

Set a breakpoint at the `calling_convention_test` function and run the program from the top (Figure 5 and 6).

```

jnelra@NEXT:~/workshop $ gdb checkparameters checkparameters.core
GNU gdb 6.1.1 [FreeBSD]
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "amd64-marcel-freebsd"...(no debugging symbols found)...
Core was generated by 'checkparameters'.
Program terminated with signal 6, Aborted.
Reading symbols from /lib/libc.so.7...(no debugging symbols found)...done.
Loaded symbols for /lib/libc.so.7
Reading symbols from /lib/ld-elf.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-elf.so.1
#0 0x0000000080095d0ea in kill () from /lib/libc.so.7
(gdb) bt
#0 0x0000000080095d0ea in kill () from /lib/libc.so.7
#1 0x0000000080095b819 in abort () from /lib/libc.so.7
#2 0x00000000004007aa in main ()
(gdb) f 2
#2 0x00000000004007aa in main ()
(gdb) i lo
no symbol table info available.
(gdb) i main
Undefined info command: "main". Try "help info".
(gdb) disas main
Dump of assembler code for function main:
0x0000000000400780 <main+0>: push %rbp
0x0000000000400781 <main+1>: mov %rsp,%rbp
0x0000000000400784 <main+4>: mov $0x400816,%edi
0x0000000000400789 <main+9>: mov $0x1,%esi
0x000000000040078e <main+14>: mov $0x2,%edx
0x0000000000400793 <main+19>: mov $0x3,%ecx
0x0000000000400798 <main+24>: mov $0x4,%r8d
0x000000000040079e <main+30>: xor %eax,%eax
0x00000000004007a0 <main+32>: callq 0x4004dc <printf@plt>
0x00000000004007a5 <main+37>: callq 0x40050c <abort@plt>
0x00000000004007aa <main+42>: nopw 0x0(%rax,%rax,1)
End of assembler dump.
(gdb)

```

Figure 3. Disas command on main

```

(gdb) disas calling_convention_test
Dump of assembler code for function calling_convention_test:
0x00000000004007b0 <calling_convention_test+0>: push %rbp
0x00000000004007b1 <calling_convention_test+1>: mov %rsp,%rbp
0x00000000004007b4 <calling_convention_test+4>: mov %ecx,%r8d
0x00000000004007b7 <calling_convention_test+7>: mov %edx,%ecx
0x00000000004007b9 <calling_convention_test+9>: mov %esi,%edx
0x00000000004007bb <calling_convention_test+11>: mov %edi,%esi
0x00000000004007bd <calling_convention_test+13>: mov $0x400816,%edi
0x00000000004007c2 <calling_convention_test+18>: xor %eax,%eax
0x00000000004007c4 <calling_convention_test+20>: callq 0x4004dc <printf@plt>
0x00000000004007c9 <calling_convention_test+25>: mov $0x3e7,%eax
0x00000000004007ce <calling_convention_test+30>: pop %rbp
0x00000000004007cf <calling_convention_test+31>: retq
End of assembler dump.
(gdb) disas main 0x04007cf
Dump of assembler code from 0x400780 to 0x4007cf:
0x0000000000400780 <main+0>: push %rbp
0x0000000000400781 <main+1>: mov %rsp,%rbp
0x0000000000400784 <main+4>: mov $0x400816,%edi
0x0000000000400789 <main+9>: mov $0x1,%esi
0x000000000040078e <main+14>: mov $0x2,%edx
0x0000000000400793 <main+19>: mov $0x3,%ecx
0x0000000000400798 <main+24>: mov $0x4,%r8d
0x000000000040079e <main+30>: xor %eax,%eax
0x00000000004007a0 <main+32>: callq 0x4004dc <printf@plt>
0x00000000004007a5 <main+37>: callq 0x40050c <abort@plt>
0x00000000004007aa <main+42>: nopw 0x0(%rax,%rax,1)
0x00000000004007b0 <calling_convention_test+0>: push %rbp
0x00000000004007b1 <calling_convention_test+1>: mov %rsp,%rbp
0x00000000004007b4 <calling_convention_test+4>: mov %ecx,%r8d
0x00000000004007b7 <calling_convention_test+7>: mov %edx,%ecx
0x00000000004007b9 <calling_convention_test+9>: mov %esi,%edx
0x00000000004007bb <calling_convention_test+11>: mov %edi,%esi
0x00000000004007bd <calling_convention_test+13>: mov $0x400816,%edi
0x00000000004007c2 <calling_convention_test+18>: xor %eax,%eax
0x00000000004007c4 <calling_convention_test+20>: callq 0x4004dc <printf@plt>
0x00000000004007c9 <calling_convention_test+25>: mov $0x3e7,%eax
0x00000000004007ce <calling_convention_test+30>: pop %rbp
End of assembler dump.
(gdb)

```

Figure 4. Disas command on calling_convention_test

Figure 5. *Using disas command*

Figure 6. Seeing machine code and register info at the same time


```
(gdb) layout asm
(gdb) layout regs
```

If you see the contents of the registers specified in the AMD64 ABI (`%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8` and `%r9`), you will see the contents of the arguments that are being passed to the function `<main+9>`. If you only want to check the registers without using the GUI, just type:

```
(gdb) info regs
```

```
(gdb) p $rcx
```

This is tougher to debug as we don't have the symbol tables available as they have been stripped from the binary. That means we do not have the function names, all references to names are gone, and we have only all the operations using memory addresses. That means we cannot even set a breakpoint using main. But really all we need to know is the address of our main function. Let's take a look at another example program: Figure 8. Compile it as usual, and strip it:

```
$ cc segfaultet.c -o segfaultet
$ strip segfaultet
```

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

void seemegodown(int,int);
int main()
{
    int a, b;
    seemegodown(a,b);
}

void seemegodown(int a, int b)
{
    int c = a + b;
    memset(&c,'H',1024);
}
```

[illegible]

Now let's find where is main at.

```
(gdb) info file
```

There is the Entry point: 0x4004e0.

Let's set a breakpoint at that memory area:

```
(gdb) b *0x4004e0
```

As we know, the program text area (code) is between `0x4004e0` and `0x4007a8`. We could disassemble the range between those addresses (not really a thing you will want to do in a real application). Another option that is nicer is to just display the program counter `$pc` the next couple of instructions every time you go down one instruction (`nexti`).

```
(gdb) display /i $pc
```

Apart from the ones described earlier, you can use these ones to take a look at what is happening in your program.


```
--Type <return> to continue, or q <return> to quit--
0x000000004007a8 - 0x000000004007b6 is .fini
0x000000004007b6 - 0x000000004007b7 is .rodata
0x000000004007b8 - 0x000000004007e4 is .eh_frame_hdr
0x000000004007e8 - 0x0000000040088c is .eh_frame
0x00000000400890 - 0x000000004008a0 is .ctors
0x000000004008a0 - 0x000000004008b0 is .dtors
0x000000004008b0 - 0x000000004008b8 is .jcr
0x000000004008b8 - 0x00000000400a28 is .dynamic
0x00000000400a28 - 0x00000000400a60 is .got.plt
0x00000000400a60 - 0x00000000400a7c is .data
0x00000000400a80 - 0x00000000400a88 is .bss

gdb) disas 0x4004e0 0x4007a8
Invalid number "0x4007a8".
gdb) disas 0x4004e0 0x4007a8
Map of assembler code from 0x4004e0 to 0x4007a8:
0x000000004004e0 <exit@plt+20>:    push    %rbp
0x000000004004e1 <exit@plt+21>:    mov     %rsp,%rbp
0x000000004004e4 <exit@plt+24>:    push    %r15
0x000000004004e6 <exit@plt+26>:    push    %r14
0x000000004004e8 <exit@plt+28>:    push    %r13
0x000000004004ea <exit@plt+30>:    push    %r12
0x000000004004ec <exit@plt+32>:    push    %rbx
0x000000004004ed <exit@plt+33>:    push    %rax
0x000000004004ee <exit@plt+34>:    mov     %rdi,%r14
0x000000004004f1 <exit@plt+37>:    mov     (%r14),%r15
0x000000004004f4 <exit@plt+40>:    movslq  %r15,%rax
0x000000004004f7 <exit@plt+43>:    lea     0x10(%r14,%rax,8),%r13
0x000000004004fc <exit@plt+48>:    cmq     $0x0,0x20057c(%rip)    # 0x600a80 <environ>
0x00000000400504 <exit@plt+56>:    jne     0x40050d <exit@plt+65>
0x00000000400506 <exit@plt+58>:    mov     %r13,0x200573(%rip)    # 0x600a80 <environ>
0x0000000040050d <exit@plt+65>:    add     $0x8,%r14
0x00000000400511 <exit@plt+69>:    test    %r15,%r15
0x00000000400514 <exit@plt+72>:    jle     0x40053b <exit@plt+111>
0x00000000400516 <exit@plt+74>:    mov     (%r14),%rax
0x00000000400519 <exit@plt+77>:    test    %rax,%rax
0x0000000040051c <exit@plt+80>:    jne     0x400523 <exit@plt+87>
0x0000000040051e <exit@plt+82>:    jmp     0x40053b <exit@plt+111>
0x00000000400520 <exit@plt+84>:    inc     %rax
0x00000000400523 <exit@plt+87>:    mov     %rax,0x200536(%rip)    # 0x600a60 <__progname>
0x0000000040052a <exit@plt+94>:    jmp     0x40053b <exit@plt+111>
0x0000000040052c <exit@plt+96>:    inc     %rax
0x0000000040052f <exit@plt+99>:    nop
0x00000000400530 <exit@plt+100>:   mov     (%rax),%cl
0x00000000400532 <exit@plt+102>:   cmq     $0x2f,%cl
0x00000000400535 <exit@plt+105>:   je      0x400520 <exit@plt+84>
0x00000000400537 <exit@plt+107>:   test    %cl,%cl
0x00000000400539 <exit@plt+109>:   jne     0x40052c <exit@plt+96>
0x0000000040053b <exit@plt+111>:   mov     $0x6008b8,%eax
0x00000000400540 <exit@plt+116>:   test    %rax,%rax
0x00000000400543 <exit@plt+119>:   je      0x40054f <exit@plt+131>
0x00000000400545 <exit@plt+121>:   mov     %esi,%rdi
0x00000000400548 <exit@plt+124>:   callq   0x40049c <atexit@plt>
0x0000000040054d <exit@plt+129>:   jmp     0x400554 <exit@plt+136>
0x0000000040054f <exit@plt+131>:   callq   0x4004ac <__init_tls@plt>
0x00000000400554 <exit@plt+136>:   mov     $0x6008b8,%eax
0x00000000400559 <exit@plt+141>:   test    %rax,%rax
0x0000000040055c <exit@plt+144>:   jne     0x400641 <exit@plt+173>
0x00000000400562 <exit@plt+150>:   mov     $0x400660,%edi
```

Figure 9. Disas command addresses range

```
0x00000000400730 <exit@plt+612>:    push    %rbp
0x00000000400731 <exit@plt+613>:    mov     %rsp,%rbp
0x00000000400734 <exit@plt+616>:    sub     $0x10,%rsp
0x00000000400738 <exit@plt+620>:    mov     -0x4(%rbp),%edi
0x0000000040073b <exit@plt+623>:    mov     -0x8(%rbp),%esi
0x0000000040073e <exit@plt+626>:    callq   0x400750 <exit@plt+644>
0x00000000400743 <exit@plt+631>:    mov     $0x0,%eax
0x00000000400748 <exit@plt+636>:    add     $0x10,%rsp
0x0000000040074c <exit@plt+640>:    pop     %rbp
0x0000000040074d <exit@plt+641>:    retq
0x0000000040074e <exit@plt+642>:    xchg    %ax,%ax
0x00000000400750 <exit@plt+644>:    push    %rbp
0x00000000400751 <exit@plt+645>:    mov     %rsp,%rbp
0x00000000400754 <exit@plt+648>:    sub     $0x10,%rsp
0x00000000400758 <exit@plt+652>:    mov     $0x48,%eax
0x0000000040075d <exit@plt+657>:    mov     $0x400,%rdx
0x00000000400767 <exit@plt+667>:    lea     -0xc(%rbp),%rcx
0x0000000040076b <exit@plt+671>:    mov     %edi,-0x4(%rbp)
0x0000000040076e <exit@plt+674>:    mov     %esi,-0x8(%rbp)
0x00000000400771 <exit@plt+677>:    mov     -0x4(%rbp),%esi
0x00000000400774 <exit@plt+680>:    add     -0x8(%rbp),%esi
0x00000000400777 <exit@plt+683>:    mov     %esi,-0xc(%rbp)
0x0000000040077a <exit@plt+686>:    mov     %rcx,%rdi
0x0000000040077d <exit@plt+689>:    mov     %eax,%esi
0x0000000040077f <exit@plt+691>:    callq   0x4004bc <memset@plt>
0x00000000400784 <exit@plt+696>:    add     $0x10,%rsp
0x00000000400788 <exit@plt+700>:    pop     %rbp
0x00000000400789 <exit@plt+701>:    retq
```

Figure 10. Here is our function (notice the call to memset in there) <exit@plt+644>

Examining memory

Use the x command to examine memory. The syntax for the x command is x/FMT ADDRESS. The FMT field is a count followed by a format letter and a size letter. Use the help command 'help x' to see them all. The ADDRESS argument can either be a symbol name, such as a variable, or a memory address.

Examine the variable as a string:

```
(gdb) x/s <var>
```

Examine the variable as a character:

```
(gdb) x/c <var>Examine the variable as 4 characters:  
(gdb) x/4c <var>
```

Examine the first 32 bits of the variable:

```
(gdb) x/t <var>
```

Examine the first 24 bytes of the variable in hex:

```
(gdb) x/3x <var>
```

Print the instructions at the memory area specified

```
(gdb) x/i <addr>
```

Stepping

We are looking at machine code now, and we don't have the source code for the commands that we usually use: next (n) and step(s) have their equivalents named with the 'l' suffix (for instruction):

- nextl (nl)
- stepl (sl)

CARLOS NEIRA

Carlos Neira has worked for several years as a C/C++ developer and on kernel porting and debugging enterprise legacy applications. He is currently employed as a C developer under Z/OS, debugging and troubleshooting legacy applications for a global financial company. He is also engaged in independent research on affective computing. In his free time, he contributes to the PC-BSD project and enjoys metal detecting.

Faster. Better. Reliable.

Trusted by over 500 ISPs worldwide.

Hyper is the first multimedia cache fully developed in Brazil, by Taghos.

With Hyper, ISPs can save on network bandwidth while increasing content-delivery speeds, resulting in end-customer satisfaction.

Features:

- 24x7x365 always-on support
- Active monitoring
- Automatic updates
- Appliance or license
- Easy deployment
- Configuration and reports via web interface



Remote Install
Using your hardware

Model	Traffic	RAM	Cache	SSD
T15	Up to 15 Mbps	8 GB	1x 1 TB	-
T50	Up to 50 Mbps	8 GB	2x 1 TB	-
T100	Up to 100 Mbps	8 GB	2x 1 TB	1x 160 GB
T150	Up to 150Mbps	16 GB	3x 2 TB	1x 160 GB
T300	Up to 300 Mbps	16 GB	5x 2 TB	1x 240 GB
T500	Up to 500 Mbps	32 GB	7x 2 TB	1x 480 GB
T1000	Up to 1 Gbps	64 GB	10x 1 TB	1x 480 GB
T2000	Up to 2 Gbps	96 GB	24x 1 TB	3x 480 GB
T3000	Up to 3 Gbps	128 GB	32x 1 TB	5x 480 GB

Visit us at www.taghos.com and start saving bandwidth today!

Getting to Grips With the Gimp – Part 8

In the eighth part in our series on the Gimp we will look at how to extend the Gimp with extra add-ons.

What you will learn...

- How to manipulate images like a design pro

What you should know...

- General PC administration skills

The Gimp currently supports a number of add-ons: Brushes, Gradients, Patterns, Palettes, Fonts, Scripts and Plug-ins. Adding an extra resource is simplicity itself, once the desired add-on has been found, download and extract it into the relevant folder under the Gimp home folder. This is located in `/home/username/.gimp-2.8` for the current version of the Gimp where username is the account you are logged in as [Figure 1].

brushes		24/08/14
curves		24/08/14
dynamics		24/08/14
environ		24/08/14
fonts		24/08/14
fractalexplorer		24/08/14
glig		24/08/14
glare		24/08/14
gimpressionist		24/08/14
gradients		24/08/14
interpreters		24/08/14
levels		24/08/14
modules		24/08/14
palettes		24/08/14
patterns		24/08/14
plug-ins		20:19
scripts		24/08/14
templates		24/08/14
themes		24/08/14
tmp		24/08/14
tool-options		09/09/14
tool-presets		24/08/14

There are far too many resources on the web to list them all, but I have compiled a list of some of the most popular in Table 1.

Table 1. Details and credits

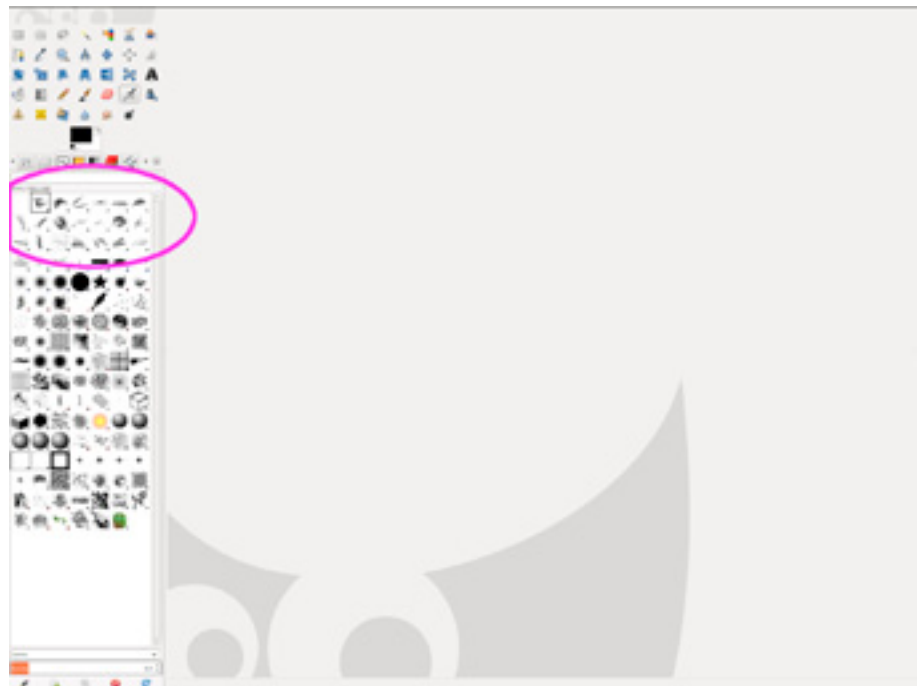
Resource	URL	Details and credit
Deviant art	http://www.deviantart.com	Massive resource of Gimp extras
Abstract fonts	http://www.abstractfonts.com	Font resource for private and commercial use
Gimp plugin registry	http://registry.gimp.org	Gimp plugins and scripts

Add a brush

Download the Aurora Borealis brush set from <http://www.deviantart.com/art/GIMP-Aurora-Brushes-71229165>.

Extract the Aurora directory from GIMP_Aurora_Brushes_by_Project_GimpBC.zip into `/home/username/.gimp-2.8/brushes` (I used Archive Manager for this).

Restart the Gimp – The new brushes will be found under the Brushes tab [Figure 2].

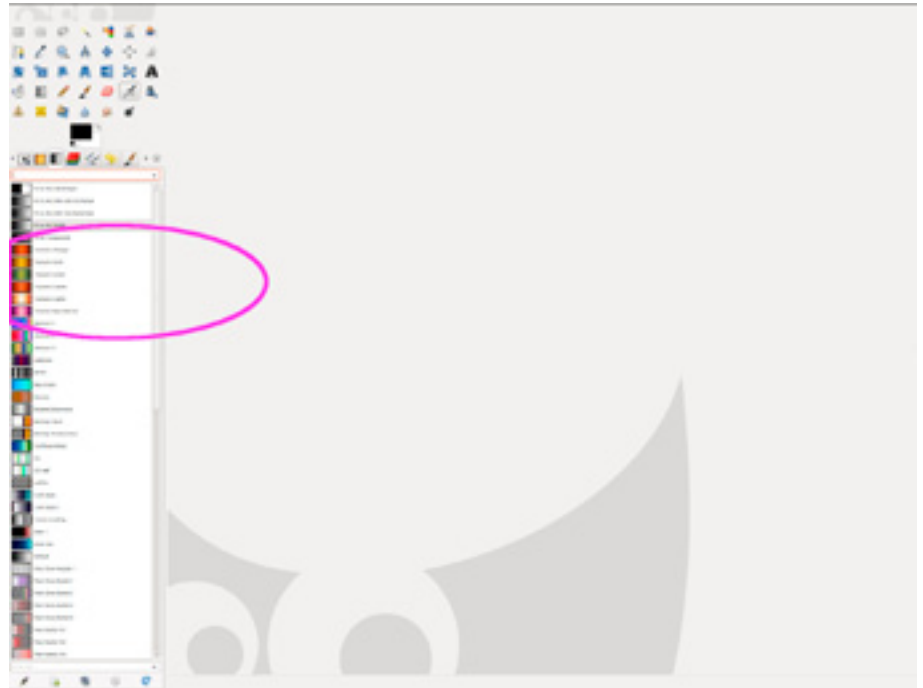


Add a gradient

Download the Autumn gradients from <http://www.deviantart.com/art/6-Autumn-Gradients-for-GIMP-187205931>.

Extract the 6 gradient files from 6_autumn_gradients_for_gimp_by_In213-d33gh0r.zip into `/home/username/.gimp-2.8/gradients`.

Restart the Gimp – The new gradient will be found under the gradient tab [Figure 3].

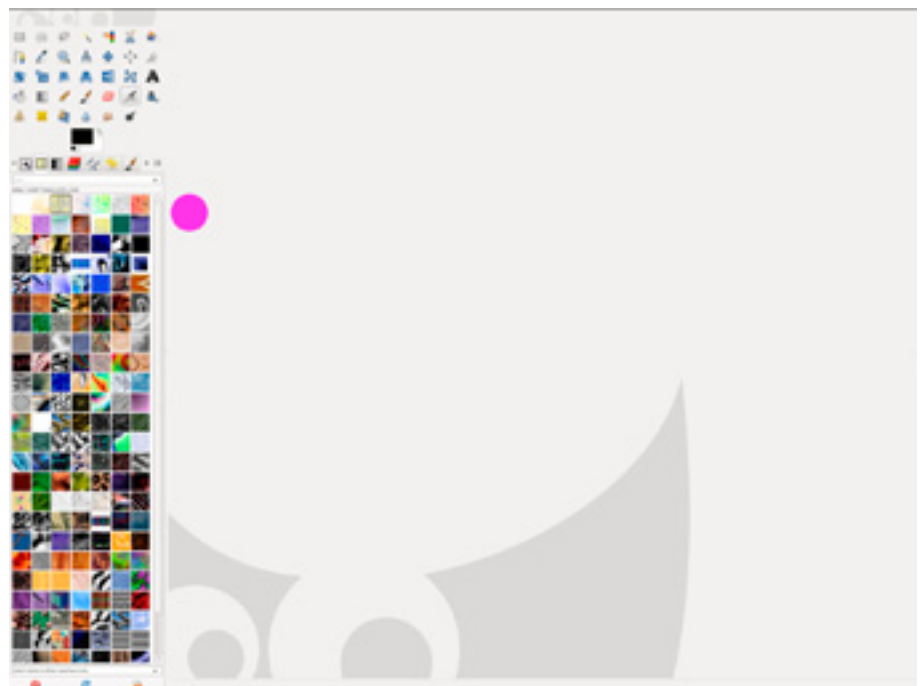


Add a pattern

Download Glitter Gimp Patterns from <http://www.deviantart.com/art/Glitter-GIMP-Patterns-187921791>.

Extract the folder and contents from glitter_gimp_patterns_by_jedania-d33vtdr.zip into `/home/username/.gimp-2.8/patterns` and delete the readme file as this will cause an error message in the Gimp.

Restart the Gimp – The new gradient will be found under the gradient tab [Figure 4].



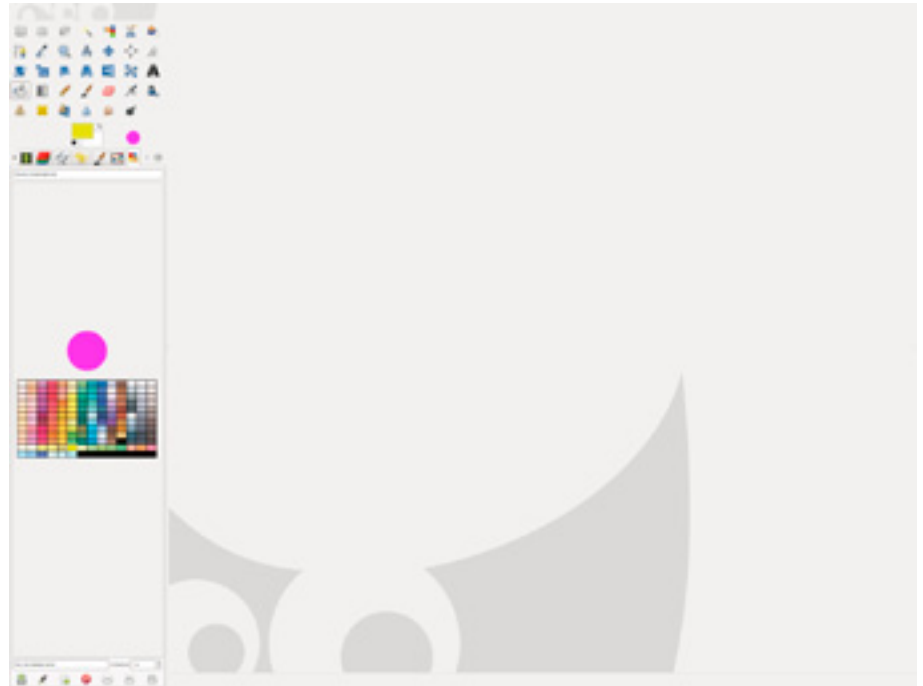
Add a palette

Download the Touch marker palette from <http://www.deviantart.com/art/Touch-marker-palette-for-GIMP-108710037>.

Extract the contents from Touch_marker_palette_for_GIMP_by_dfmurcia.zip into `/home/username/.gimp-2.8/palettes`.

Ensure the palettes tab is enabled via Windows → Dockable dialogues → Palettes.

Restart the Gimp – The new palette will be found under the palettes tab [Figure 5].

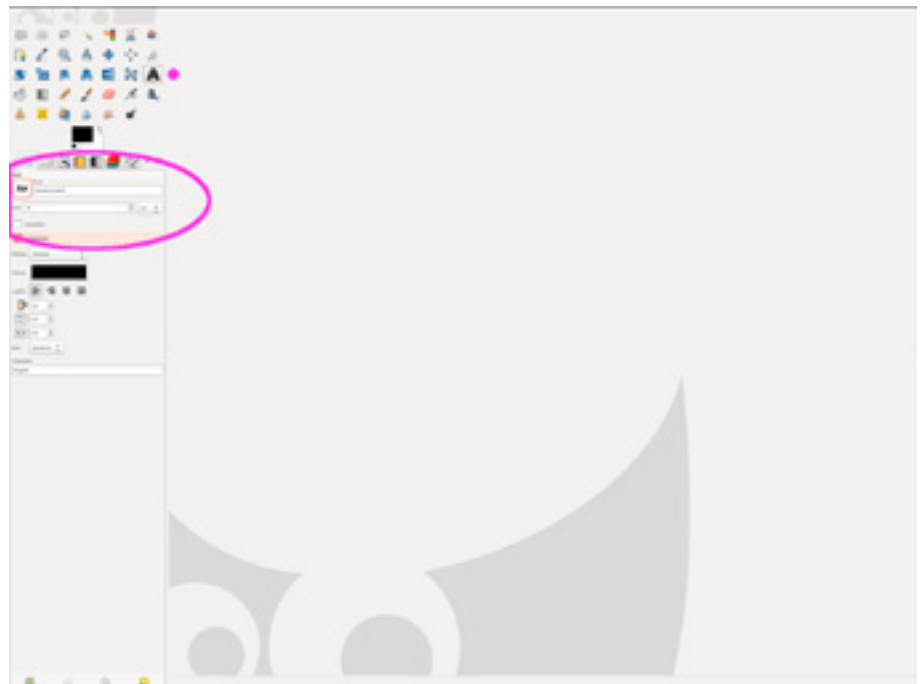


Add a font

Download Harabara regular from <http://www.abstractfonts.com>.

Extract the TTF file from harabara_regular.zip into `/home/username/.gimp-2.8/fonts`.

Restart the Gimp and the font will be available under the text button [Figure 6].

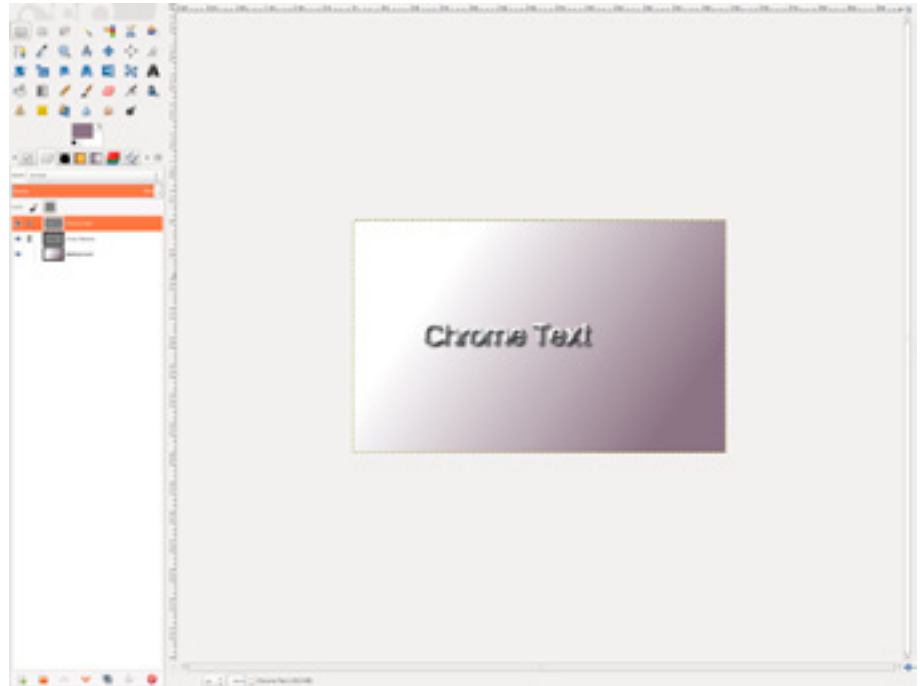


Add a script

Download the Gimp Chrome script by Fence-post from <http://www.deviantart.com/art/GIMP-Chrome-Scripts-77288538> into the `/home/username/.gimp-2.8/scripts` directory.

Refresh scripts by going to Filters → Script-Fu → Refresh scripts.

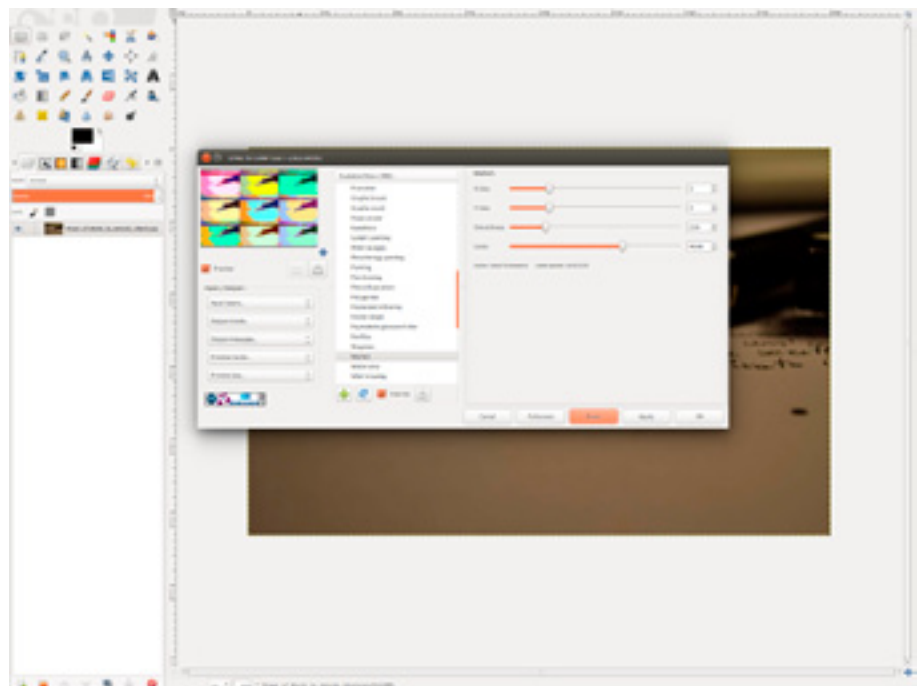
Add some text to a new image, and apply the filter via Filters → Alpha to logo → Fencepost chrome [Figure 7].



Add a plug-in

Download the G'MIC from <http://registry.gimp.org/node/13469> and extract the content of the zip file to `/home/username/.gimp-2.8/plugins` directory.

Restart the Gimp, load an image of your choice and run the plugin from Filters → G'MIC. You will now have an additional 400 effects you can apply to your image [Figure 8].



ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

\$99.95

The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
User Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD



FreeBSD 9.1 Jewel Case CD/DVD \$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD \$39.95

FreeBSD 9.0 DVD \$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 \$29.95

FreeBSD Subscription, start with DVD 9.1 \$29.95

FreeBSD Subscription, start with CD 9.0 \$29.95

FreeBSD Subscription, start with DVD 9.0 \$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD \$29.95

PC-BSD Subscription \$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) \$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide) \$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) \$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 \$79.95

PC-BSD 9.0 Users Handbook \$24.95

BSD Magazine \$11.99

The FreeBSD Toolkit DVD \$39.95

FreeBSD Mousepad \$10.00

FreeBSD & PCBSD Caps \$20.00

BSD Daemon Horns \$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even MORE items
visit our website today!

www.FreeBSDMall.com

The UK government is planning to put trolls in jail for up to two years. Is this a sensible approach in containing the darker side of human nature?

Trolling, the deliberate posting of offensive or controversial comments via Forums, Facebook, Twitter etc. is not a new phenomena on the big nasty inter-web. The phrase “Don’t feed the trolls” (DFTT) has been around as long as I remember – going back to the 1200 baud modem and bulletin boards in fact. But what is the psychology behind this desperate and compulsive act? Apart from the nastier incidents like threats, bullying and the tragic case where a grieving family was sent pictures of their deceased daughter after an accident, most people would agree that trolls are either immature, borderline psychopaths and sociopaths, bad mannered, deeply insecure or a combination of all of these characteristics. The nastier breed are just evil. But is this sufficient reason to newly criminalise behaviour that traditionally was dealt as a civil matter with damages being awarded? While physical abuse has always been dealt with under criminal law, character assassination, defamation, slander and libel are covered by civil law.

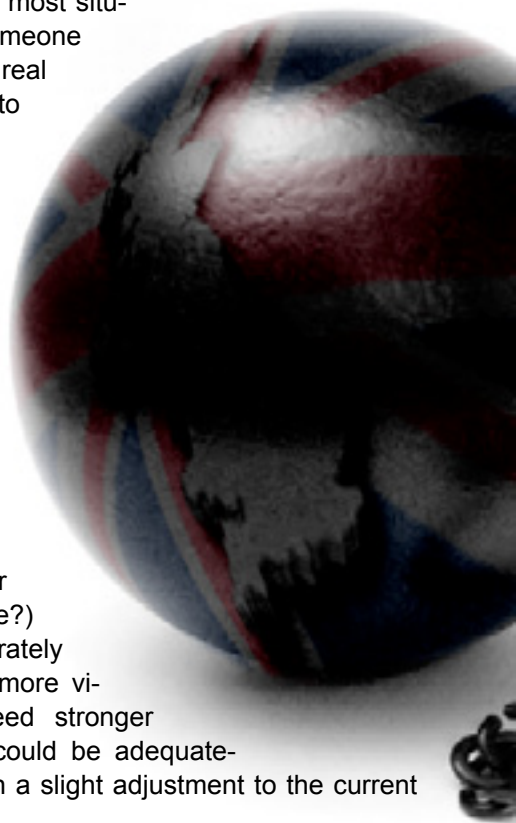
Studies have shown that individuals engage very differently online than in “real life” or face to face situations. The current debate about online ethics and behaviour questions whether online activity is closer to reality than fantasy – e.g. you should not engage in behaviour online than you would not do in normal physical life. While I agree with this statement up to a point, the online boundaries are blurred between reality and fantasy. What is appropriate in a online game session, the sports field or the workplace are very different standards indeed. Human beings adapt to their environment, and the surrealism of the Internet does not help us easily identify where the moral boundaries are. I would go as far as saying that peoples personality subtly changes when they go online, very much like the quiet man becomes a speed freak or road rage instigator when behind the wheel of a car.

Anonymity is a two edged sword – from a psychological standpoint people are quickly dehumanised by it, so it is easy to rationalise that there is not a flesh and blood person at the end of that Twitter account or what-

ever. That is the real problem with the troll – while their actions may be considered sport, a bit of leg pulling, or based on getting a rise out of people reactions, they probably don’t care about the recipients feelings, but if they do it is a much more pernicious attack – something akin to a damaged child pulling the wings off flies or torturing cats. These are the true psychopath trolls rather than the less malevolent irritants.

So what can we do about trolling? The old adage “Don’t feed the trolls” is probably adequate in most situations. After all, if someone is pulling your leg in real life you would tend to ignore them unless you wanted to engage in the joke. You might even deliver a swift verbal rebuke, but that just demonstrates that you have taken the bait. A Zen like calm is the best approach as it starves the lesser troll of the oxygen (or is it sulphur dioxide?) that they so desperately need. However, the more vicious variety do need stronger sanctions, and this could be adequately covered in law with a slight adjustment to the current stalking laws.

This route however, completely misses the point about network abuse. There are many facets of this – scamming, hacking, phishing, spamming and of course trolling. What we really need is a modification and strength-



ening of the Malicious Communications Act to include the former 4 actions. Ironically, the act currently covers trolling. To quote: 'The Malicious Communications Act 1988 deals with the sending to another of any article which is indecent or grossly offensive, or which conveys a threat, or which is false, provided there is an intent to cause distress or anxiety to the recipient. The offence covers letters, writing of all descriptions, electronic communications, photographs and other images in a material form, tape recordings, films and video recordings.' So in regards to trolling we have the situation (like the phone hacking scandal) where legislation already exists but is not enforced. The political solution to this? Write up another law.

To me, this appears as a typical political knee-jerk reaction while at the same time ignoring the bigger picture. While worse case a vicious troll may drive an individual to suicide (and there is no excuse for the full weight of the law not to be exercised in such instances), in the majority of cases all we have is hurt feelings and a few blows to our pride. The other four forms of lowlife cause much more damage both financially and to reputations. Pity the poor student who has his dissertation permanently encrypted by Cryptowall or a business that has its bank account emptied and livelihoods are destroyed. We seem to have lost the plot where political correctness and hate crime take priority over the nastier network crimes, especially taking into account the sheer

scale of the problem. You might encounter a few trolls per week, but the other forms of abuse are legion.

So it looks like once again the powers that be are disconnected from the reality of the Internet. Two years is a long time to spend in prison for a wind up, and the cost to John Q Taxpayer (and society) is large. Prosecution and police costs are estimated to be £65K with the annual cost of keeping a prisoner in jail £40K. For a maximum sentence, that would cost £145K. Compare that with the cost of a civil action to bankrupt a troll, and depending on their circumstances, the possibility that the troll can repay their victim in cold hard cash and I know which judicial option my wallet prefers. In this time of austerity, we need to prioritise – cut our cloth according to what we have. To create new law at great cost while ignoring bigger issues and at the same time not using the laws we have at our disposal is wasteful, and ethically more offensive that the problem it is trying to address.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.





Dr.Web 9.0

for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search



© Doctor Web
2003 — 2013

www.drweb.com

Free 30-day trial: <https://download.drweb.com>

New features in Dr.Web 9.0 for Windows: <http://products.drweb.com/9>

FREE bonus — Dr.Web Mobile Security:
<https://download.drweb.com/android>



Among clouds Performance and Reliability is **critical**



Download syslog-ng Premium Edition
product evaluation [here](#)

Attend to a free logging tech webinar [here](#)



BalaBit
IT Security

www.balabit.com

syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the
Reliable Log Transfer Protocol™
- trusted log transfer and storage